Operations management at container terminals using advanced information technologies

Elisabeth Zehendner

NNT : 2013 EMSE 0714

THÈSE

présentée par

Elisabeth ZEHENDNER

pour obtenir le grade de
Docteur de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Génie Industriel

OPERATIONS MANAGEMENT AT CONTAINER TERMINALS USING ADVANCED INFORMATION TECHNOLOGIES

GESTION DES OPÉRATIONS DANS LES TERMINAUX À CONTENEURS À L'AIDE DE TECHNOLOGIES DE L'INFORMATION AVANCÉES

soutenue à Gardanne, le 23 octobre 2013

Membres du jury

Président :	Patrick JAILLET	Professeur, Massachusetts Institute of Technology, Boston, États-Unis
Rapporteurs :	Rommert DEKKER	Professeur, Erasmus University Rotterdam, Rotterdam, Pays-Bas
	Eric SANLAVILLE	Professeur, Université du Havre, Le Havre, France
Examinateurs :	Pierre CARIOU	Professeur associé, Euromed Management, Marseille, France
	Stéphane DAUZÈRE-PÉRÈS	Professeur, ENSM-SE, Gardanne, France
Directeur de thèse :	Dominique FEILLET	Professeur, ENSM-SE, Gardanne, France

Spécialités doctorales :	Responsables :
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche
MECANIQUE ET INGENIERIE	S. Drapier, professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche
MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
INFORMATIQUE	O. Boissier, Professeur
IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
GENIE INDUSTRIEL	A. Dolgui, Professeur
MICROELECTRONIQUE	

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BENABEN	Patrick	PR1	Sciences et génie des matériaux	CMP
BERNACHE-ASSOLLANT	Didier	PR0	Génie des Procédés	CIS
BIGOT	Jean Pierre	MR(DR2)	Génie des Procédés	SPIN
BILAL	Essaid	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR1	Informatique	FAYOL
BORBELY	Andras	MR(DR2)	Sciences et génie de l'environnement	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BURLAT	Patrick	PR2	Génie Industriel	FAYOL
COURNIL	Michel	PR0	Génie des Procédés	DIR
DARRIEULAT	Michel	IGM	Sciences et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Image Vision Signal	CIS
DELAFOSSE	David	PR1	Sciences et génie des matériaux	SMS
DESRAYAUD	Christophe	PR2	Mécanique et ingénierie	SMS
DOLGUI	Alexandre	PR0	Génie Industriel	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FEILLET	Dominique	PR2	Génie Industriel	CMP
FOREST	Bernard	PR1	Sciences et génie des matériaux	CIS
FORMISYN	Pascal	PR0	Sciences et génie de l'environnement	DIR
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Génie des Procédés	SPIN
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GIRARDOT	Jean-jacques	MR(DR2)	Informatique	FAYOL
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HAN	Woo-Suck	CR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
INAL	Karim	PR2	Microélectronique	CMP
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
LI	Jean Michel		Microélectronique	CMP
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MONTHEILLET	Franck	DR	Sciences et génie des matériaux	SMS
PERIER-CAMBY	Laurent	PR2	Génie des Procédés	DFG
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR1	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Image Vision Signal	CIS
POURCHEZ	Jérémy	CR	Génie des Procédés	CIS
ROUSTANT	Olivier	MA(MDC)		FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
SZAFNICKI	Konrad	MR(DR2)	Sciences et génie de l'environnement	CMP
TRIA	Assia		Microélectronique	CMP
VALDIVIESO	François	MA(MDC)	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	MR(DR2)	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR0	Génie industriel	CIS

ENISE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

BERGHEAU	Jean-Michel	PU	Mécanique et Ingénierie	ENISE
BERTRAND	Philippe	MCF	Génie des procédés	ENISE
DUBUJET	Philippe	PU	Mécanique et Ingénierie	ENISE
FORTUNIER	Roland	PR	Sciences et Génie des matériaux	ENISE
GUSSAROV	Andrey	Enseignant contractuel	Génie des procédés	ENISE
HAMDI	Hédi	MCF	Mécanique et Ingénierie	ENISE
LYONNET	Patrick	PU	Mécanique et Ingénierie	ENISE
RECH	Joël	MCF	Mécanique et Ingénierie	ENISE
SMUROV	Igor	PU	Mécanique et Ingénierie	ENISE
TOSCANO	Rosario	MCF	Mécanique et Ingénierie	ENISE
ZAHOUANI	Hassan	PU	Mécanique et Ingénierie	ENISE

Mise à jour : 07/04/2013

PR 0	Professeur classe exceptionnelle
PR 1	Professeur 1ère classe
PR 2	Professeur 2ème classe
PU	Professeur des Universités
MA (MDC)	Maître assistant
DR	Directeur de recherche

Ing.	Ingénieur
MCF	Maître de conférences
MR (DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
IGM	Ingénieur général des mines

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
FAYOL	Institut Henri Fayol
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Acknowledgments

I would like to express my gratitude to my supervisor Dominique Feillet who supported, encouraged and guided me throughout my thesis. I especially thank him for sharing his vast knowledge, for his contributions to my thesis, for his proof-reading and for his overwhelming availability.

I would like to thank the entire SFL team for the nice reception and the pleasant time spent at work and outside. Special thanks go to Stéphane Dauzère-Pérès, Nabil Absi and Gloria Rodriguez Verjan for their productive cooperation, to Jakey Blue for his help on statistical methods and to Sebastian Knopp for his advice on C++.

I would like to express my deep gratitude to Patrick Jaillet, Cynthia Barnhart, Vahideh Manshadi and Setareh Borjian who invited me to Boston to work with them. I highly appreciate their guidance on dynamic optimization.

I owe deep gratitude to Gérard Bricout, Frédéric Pellegrin and Yann Tessier from Seayard and to Christophe Reynaud from Marseille Gyptis International. They provided invaluable information on the organization, processes and problems at container terminals.

I would also like to thank all participants of the EURO Summer Institute on Maritime Logistics for the fruitful discussions. Finally, I would like to thank my family and friends for their support during this time and beyond.

Contents

Chapter 1

General introduction

"Containerization - the stowage of regularly or even irregularly shaped freight in sealed, reusable boxes with standardized dimensions - is one of the most important cargo-moving techniques developed in the 20th century. Being highly efficient, it has influenced and revolutionized not only the shipping industry and ports, it has also fundamentally changed the whole international trade as well as concept, design, functions and activities of transport systems in the world." (Stahlbock and Voß; 2008b)

This chapter provides general information about maritime containerized transportation and an overview of the work presented in this thesis. Section 1.1 introduces the concept of containerization, its development over the last few decades and its main players. Section 1.2 describes the structure of container terminals and their handling equipment. Section 1.3 presents optimization problems arising at container terminals. Section 1.4 illustrates the use of new technologies to improve the efficiency of container terminals. Section 1.5 states the problems dealt with in this thesis and details the structure of the thesis.

1.1 Containerized transportation

A container is a "box to carry freight, strong enough for repeated use, usually stackable and fitted with devices for transfer between [transport] modes" (Economic Commission for Europe; 2001). Standardized containers are 20 or 40 feet long and play a major role in intermodal transport where goods are moved "in one and the same loading unit [...], which uses successive two or more modes of transport [(road, rail, water)] without handling the goods themselves in changing modes" (Economic Commission for Europe; 2001). A container may for example be transported to the harbor by truck, from one harbor to another by vessel and from the harbor to its destination by train. Besides efficient discharging and loading processes, containers also improve and simplify scheduling and controlling and serve as protection against weather and pilferage (Steenken et al.; 2004).

Over the last thirty years, container shipping has grown fifteenfold from 102 millions of tons loaded in 1980 to 1 498 millions of tons loaded in 2012. In 1980, it accounted for 2.7% of total seaborne trade (measured in tons loaded) and in 2012 for 16.1% (UNCTAD Secretariat; 2012). Figure 1.1 - taken from UNCTAD Secretariat (2012) - shows the global container trade from 1996 to 2013 measured in TEUs (twenty foot equivalent) and the annual percentage change. Container shipping has grown with an impressive average annual rate of 5% to 15% over the last fifteen years. Only during the crisis in 2009 containerized

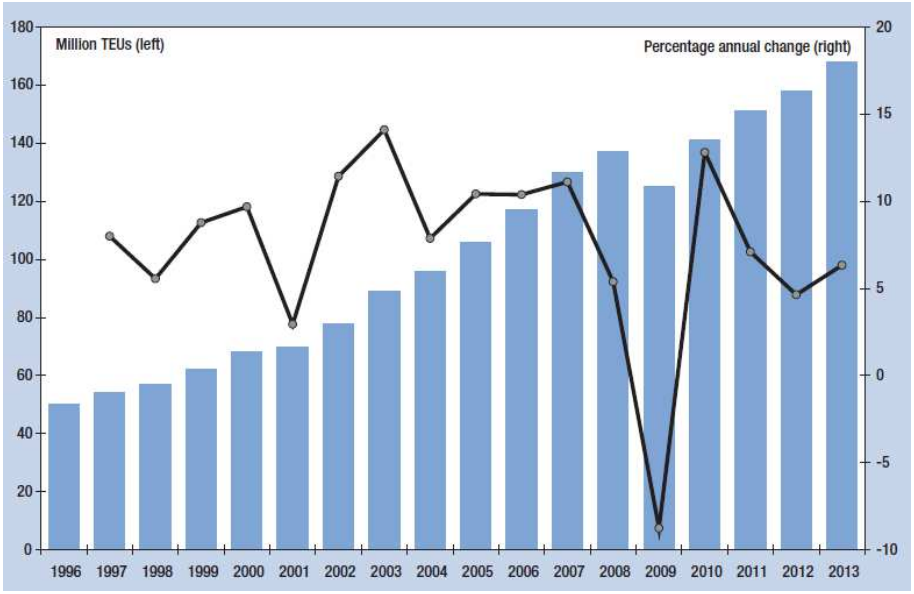


Figure 1.1: Global container trade from 1996 to 2013

transportation dropped by 10%, but recovered in 2010 to 2012. To keep pace with the tremendous increase in containerized cargo, the number and size of container ships grew continuously. The total TEU capacity of container ships has increased from 1.2 million TEU in 1987 to 12.8 million TEU in 2010. The average capacity per vessel has increased from 1 155 TEU in 1987 to 2 742 TEU in 2010 (UNCTAD Secretariat; 2010). The largest container ships in service in early 2013 have a nominal capacity of 16 000 TEU (CMA-CGM).

Containerized transport is dominated by few big shipping lines. The top 20 liner shipping companies operate 69.6% of the world total TEU capacity and the top 10 companies 51.6% (UNCTAD Secretariat; 2012). Due to the high market concentration and high vessel operating costs, the competitiveness of a container terminal depends on fast vessel turnaround times and low rates for loading and discharging containers. Other performance parameters such as gate utilization, container dwell time and the idle rate of equipment are secondary.

Recently, the connection of a container terminal to its hinterland gained in significance. Since geographical close ports may serve similar inland areas, terminals offering short and reliable delivery times have an advantage over their competitors. Figure 1.2 - taken from Notteboom (2008) - shows European container terminals and their overlapping hinterlands. According to Notteboom and Winkelmans (2004), potential cost savings in the shipping industry are getting smaller and the pressure to find cost savings elsewhere is growing. Inland transport accounts for 40% to 80% of total costs of intermodal container shipping and offers thus huge possibilities to reduce overall transportation costs.

1.2 Container terminals and their handling equipment

Container terminals transfer containers between sea vessels and inland transport modes (trucks, trains and barges). Three types of container flows are distinguished: *import containers* that arrive on vessels and leave on inland transport modes, *export containers* that arrive on inland transport modes and leave on vessels and *transshipment containers* that

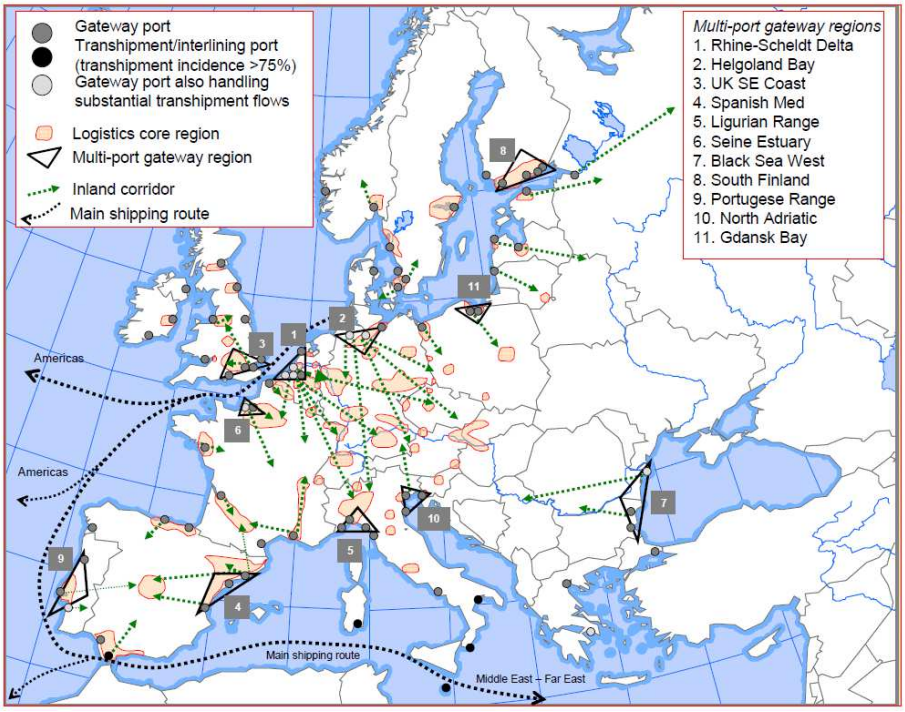


Figure 1.2: European container terminals and logistics core regions in the hinterland

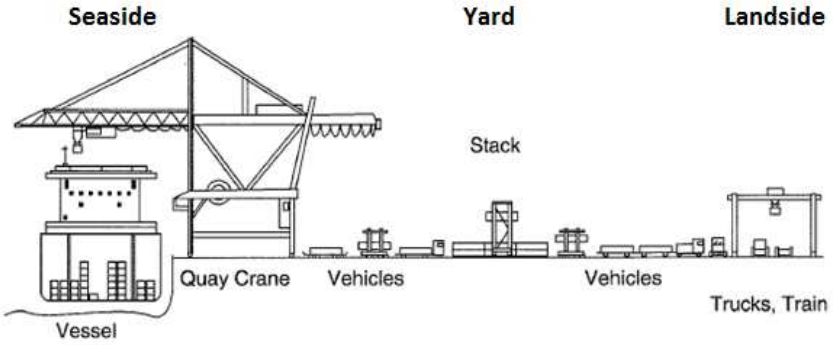


Figure 1.3: Schematic side view of a container terminal

arrive and leave on vessels. Container terminals also handle specific container types like hazardous, reefer and empty containers.

The terminal can roughly be divided into three areas: the seaside, the landside and the yard. The *seaside* is the terminal's interface with the maritime transportation system; the *landside* the interface with the inland transportation system; the *yard* serves as a temporary storage location for full and empty containers. It decouples (un)loading operations at the seaside and the landside.

Figure 1.3 - taken from Steenken et al. (2004) - depicts a terminal with its three areas and the handling equipment that may be used for each area. At the seaside, *quay cranes* load and unload vessels (see Figure 1.4). For internal transportation, different transport vehicles can be used: straddle carriers, trucks with trailers and automated guided vehicles (AGVs) are the most common ones. *Straddle carriers* (see Figure 1.5) are man-driven vehicles that are able to pick up a container at its origin, transport it and put it down at its destination. Since

Figure 1.4: Quay cranes loading a vessel

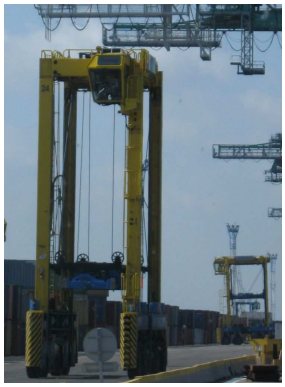


Figure 1.5: Straddle carrier

Figure 1.6: Seaside transportation via AGVs

Figure 1.7: Yard run with straddle carrier

Figure 1.8: Yard run with rail mounted gantry cranes

recently, few terminals use automatic straddle carriers (ASC) instead of manned straddle carriers. *Trucks with trailers* and *automated guided vehicles* (see Figure 1.6 - taken from Steenken et al. (2004)) transport containers from their origins to their destinations. But, they cannot lift containers on their own and have to be (un)loaded by cranes. Until now, AGVs are only used for seaside transportation.

Most terminals stack containers on the ground due to space restrictions[1]. Straddle carriers or yard cranes are used to execute storage and retrieval tasks. Straddle carriers can stack containers up to 4 tiers high. In this case, straddle carriers also execute transportation tasks. The two most common *yard cranes* are rail mounted gantry cranes (RMG) and rubber tired gantry cranes (RTG). They span 8 to 12 rows of containers and stack containers 4 to 10 tiers high. In this case, transportation tasks are executed by trucks with trailers, AGVs or manned straddle carriers. This requires a strong coordination between cranes and transport vehicles to fully use the available capacity. Figure 1.7 shows a yard run with straddle carriers; Figure 1.8 - taken from Koppe and Brinkmann (2008) - a yard run with rail mounted gantry cranes.

Different equipment is used to (un)load landside transport modes. Trucks may be (un)loaded directly in the yard by the yard crane or at a specific exchange area via straddle carriers; trains by dedicated cranes, fork lifts or reach stackers; and barges by dedicated cranes or by the same cranes than vessels. Empty containers are usually handled by reach stackers. Reefer and hazardous containers are handled with the same equipment as other containers, but might be assigned to specific areas within the yard.

The chosen terminal layout and equipment highly influence the way terminals work. This choice depends on the characteristics of the terminal. Yard cranes use the available land more efficiently since containers are stacked higher. Straddle carriers provide more flexibility. Automated terminals using yard cranes and AGVs reduce manned labor but require high initial investments. Several studies present different handling equipment and terminal layouts and evaluate the impact on the terminal performance (e.g., Ioannou et al.; 2000; Vis; 2006; Ioannou and Jula; 2008; Brinkmann; 2011; Wiese et al.; 2011; Kemme; 2013).

1.3 Optimization problems at container terminals

Increasing volumes and vessel sizes, space restrictions and severe competition put pressure on container terminals to manage their resources efficiently. In addition, terminals have to comply with environmental, congestion and labor force restrictions. This need for efficiency promulgated the use of information technologies and optimization methods. It also boosted academic research on container terminal operations since the mid-nineties (Woo et al.; 2011). Several papers classify optimization problems at container terminals and summarize related literature (e.g., Vis and de Koster; 2003; Steenken et al.; 2004; Stahlbock and Voß; 2008a; Kim; 2008). We report their classification and point out review articles for further information on the different problems.

To plan the service of vessels, several topics have to be addressed: stowage planning, berth allocation and crane split. *Stowage planning* consists in assigning containers to slots

[1] Alternatively, containers are parked on chassis that are moved with trucks.

in the ship based on container attributes (like container type, destination and weight) and to ensure the stability of the vessel. *Berth allocation* (BAP) allocates arriving vessels to the berth based on vessel specific data (like length, draft, expected arrival and service times). It provides an allocation in time and space. *Crane split* assigns quay cranes to vessels (QCAP) based on the volume to be (un)loaded and schedules single tasks on the assigned cranes (QCSP) respecting precedence constraints imposed by the stowage plan. These problems may be solved separately or in an integrated way. More information on seaside operation planning and related literature is given by Meisel (2009) and Bierwirth and Meisel (2010).

Incoming containers are not immediately loaded on an outgoing vehicle, but stored in the yard for up to several days. Terminals stack containers to use their scarce land efficiently. Consequently, they can access only the topmost container of each stack directly. If another container has to be retrieved, containers above have to be relocated. These unproductive moves cannot be avoided completely as little information about future retrievals is known when a container has to be stored. *Yard optimization* aims to minimize storage and retrieval times of containers to improve the overall performance of the terminal. It encompasses two main problems: where to store an incoming container (*storage space allocation problem*) and how to relocate containers to avoid further relocations (*remarshalling*, *premarshalling* and *container relocation problem*). Caserta, Schwarze and Voß (2011) provide more information on yard optimization and related literature.

Transport optimization deals with the transport of containers between the yard and the seaside or landside. The terminal has to decide how to allocate the transport equipment to the different tasks and how to schedule containers on the allocated equipment. The objective is to minimize waiting times of quay cranes and external transport vehicles and to reduce travel times of internal transport vehicles. Stahlbock and Voß (2008b) present a literature review on seaside and landside transport.

All these problems are highly interrelated. The quay crane throughput, for example, is related to the arrival rate of containers during the loading process and departure rates during unloading. These rates depend themselves on transport operations, but also on storage position. In addition, external and internal vehicles cause uncertainty via delays or break downs. To accurately represent the dynamic and interrelated character of container terminals *simulation* is used. At a strategic level, simulation compares different terminal layouts and types of handling equipment. At an operational and tactical level, it evaluates optimization methods. Angeloudis and Bell (2011) review simulation models designed for container terminals.

1.4 Application of intelligent freight technologies

Intelligent freight technologies are technologies that monitor and manage physical assets and information flows. Container terminals use intelligent freight technologies to exchange data with shipping lines, trucking companies and customs, to locate containers and equipment within the terminal and to automate identification tasks (Wolfe and Troup; 2005; Morais and Lord; 2006; Tsilingris et al.; 2007). The main benefits are an optimized decision making and enhanced security.

The obtained data provides information on the expected arrival of vessels, on containers to be (un)loaded, on empty positions in the yard and on current positions of containers and

of internal handling equipment. Optimization methods and the terminal operating system (TOS) provide instructions (e.g., where to store a container, which container to transport next) based on this data. Without a detailed knowledge about the current situation at the terminal, optimization methods cannot provide good solutions. Reliable data and optimization methods are essential for automated container terminals where AGVs and automated stacking cranes are managed by the system. Intelligent freight technologies are also used to automate container and truck identification at the gate and within the terminal. This makes it possible to automate the gate process, to avoid mix up of containers in the yard and to complicate fraud.

Optical character recognition (OCR), differential global positioning system (DGPS) and radio frequency identification (RFID) are used to identify trucks and containers at the terminal gate and/or to track vehicles and movements of containers within the yard. OCR converts scanned images of text into machine-encoded text. It does not need sophisticated hardware but its applicability depends on the quality of the text to be recognized. It is used to read the container ID or truck plate at the terminal gate to automate the gate process. It is also installed on container handling equipment in the yard to automatically check if the right container is handled.

DGPS is an enhanced GPS that provides greater location accuracy than GPS. Container terminals use DGPS to register yard positions of containers. DGPS receivers are not placed on containers, but on transport and stacking equipment. Every time a container is stored or retrieved, the position is measured and transmitted. This enables the software to keep track of container positions. DGPS has low infrastructure costs, but quay cranes and container stacks interfere with the signals which may lead to inaccuracy.

RFID systems consist of tags and readers. Tags are electronic chips encoded with data, readers access the data encoded on the tag. RFID provides identification without requiring line of sight, can read at short and long distance and can transmit significant amounts of data. At the terminal gate, RFID tags are used to identify trucks and the software then determines the associated container. To determine the location of containers or equipment within the terminal, RFID is embedded in a real time location system (RTLS). The position of tagged equipment is determined with the help of several RFID readers and software computing the position. Like for DGPS, container positions are transmitted every time a container is handled.

Up to now containers are not commonly tagged with RFID tags. This is due to the huge number of containers in use and to the fact that different parties (shipping lines, trucking companies, terminal operators) have to find a consensus as individual solutions applied at only one terminal are not feasible. Only recently, some containers are tagged with RFID seals to alert of tampering. The ISO 18000 standard related to RFID for item management was published some years ago which may help to develop a network for tracking containers. RFID tags could also be equipped with sensors and GPS to constantly report the location of the container and the conditions within it. Current trends on RFID for containerized transportation are reported in Dempsey (2011) and dedicated journals (RFID Journal[2], Port Technology International[3]).

[2]www.rfidjournal.com

[3]www.porttechnology.org

1.5 Structure of the thesis

Part I: Straddle carrier allocation problem

The first part of the thesis uses information on announced volumes to allocate internal handling equipment. It applies to container terminals serving different transport modes and using manned straddle carriers. Our objective is to provide the terminal operator with a tool to 1) estimate the number of straddle carriers needed for the next day to handle the announced workload and 2) to propose a possible allocation of these straddle carriers to trucks, trains, barges and vessels to minimize overall delays at the terminal.

Parts of this work (Chapter 4) have been done during my master thesis in cooperation with Nabil Absi, Stéphane Dauzère-Pérès and Gloria Rodriguez Verjan from Ecole des Mines de Saint-Etienne. Parts of this work have been published in Zehendner et al. (2011), Zehendner and Feillet (2013) and Zehendner et al. (2013) and presented at different conferences (ICCL 2011, LOGMS 2012, ROADEF 2011, ROADEF 2012).

Chapter 2: Straddle carrier allocation problem (SCAP)

This chapter introduces the straddle carrier allocation problem which has not been addressed in scientific literature yet. We discuss different strategies that may be applied to serve different transport modes and introduce a notation to describe them. We also summarize literature on resource allocation and task-scheduling problems at container terminals.

Chapter 3: Mixed integer program for SCAP

This chapter represents the straddle carrier allocation problem as a network flow problem: containers to be moved are modeled as flows and allocated straddle carriers as arc capacities. We implement the flow problem as a linear mixed integer program: we formulate a generic core model and extend it to the service strategies described in the previous chapter. We also formulate an aggregated model that reduces the number of variables, but may be used only for few service strategies. We discuss the complexity of the extended models and carry out a sensitivity analysis to determine the impact of different input parameters.

Chapter 4: Case study: Grand Port Maritime de Marseille

This chapter presents a case study carried out for a container terminal at the Grand Port Maritime de Marseille. We formulate the model for the different service strategies applied at this terminal and conduct experiments on real-world instances. We show via simulation that results obtained by the deterministic optimization problem remain valid in an uncertain environment. We do not investigate advanced solution methods, since these real-world instances can be solved by a commercial solver very quickly.

Chapter 5: SCAP and truck appointment systems

This chapter combines the straddle carrier allocation problem with the dimensioning of a truck appointment system. The objective is to use the truck appointment system to deviate truck arrivals to less busy periods to reduce overall delays at the terminal. We adapt the model from Chapter 3 to simultaneously determine the number of truck appointments to offer

and the straddle carrier allocation. Experiments evaluate the impacts of the appointment system via optimization and simulation.

Part II: Container relocation problem

The second part of the thesis uses information on announced container retrievals and container positions to improve retrieval operations. The problem we deal with is known as the container relocation problem. The objective is to retrieve containers in a given sequence with a minimum number of parasite relocations. Like most other studies, we consider the case where only containers above the target container may be relocated.

Parts of this work (Chapter 10) have been done in cooperation with Patrick Jaillet, Cynthia Barnhart, Vahideh Manshadi and Setareh Borjian from Massachusetts Institute of Technology. Intermediate results have been presented at different conferences (IMHRC 2012, OR 2012, ROADEF 2013).

Chapter 6: Container relocation problem (CRP)

This chapter introduces the container relocation problem and summarizes related literature. It also presents existing bounds on the number of relocations and commonly used instances. We also introduce a new upper bound on the number of relocations.

Chapter 7: Binary integer program for CRP

This chapter presents an existing binary integer programming model. We then improve the model formulation by correcting two errors and by omitting superfluous variables. We also present a preprocessing mechanism to fix variables and introduce cuts. Computational results evaluate the impact of the preprocessing mechanism and of cuts.

Chapter 8: Branch and price approach for CRP

The binary model from the previous chapter is impractical for bigger instances. This chapter presents a branch and price approach to solve bigger instances. We present the decomposition of the binary formulation into a master problem and a subproblem as well as a branching strategy. We introduce a new bound on the number of relocations based on values of dual variables. We also present two alternative subproblems based on enumeration together with two mechanisms to reduce the number of relocations. Computational experiments evaluate the performance of the different approaches.

Chapter 9: Heuristic branch and price approach for CRP

Due to the huge number of feasible columns, the enumerative subproblem does not generate columns quickly. This chapter presents a heuristic subproblem to quickly generate new columns by solving a network flow problem. The heuristic column generation is embedded in a branching procedure. The objective is to determine a good integer solution rather than an optimal fractional solution. New integer solutions are obtained by running a heuristic based on columns determined by the subproblem. Experiments evaluate different parameter settings and compare the heuristic branch and price to existing approaches.

Chapter 10: Dynamic container relocation problem

Until now, we assumed that the entire retrieval sequence is known in advance. This is not realistic for truck arrivals. This chapter deals with a dynamic and more realistic version of the container relocation problem, where information about container retrievals becomes revealed over time. We summarize related literature and introduce a criterion to evaluate the quality of a given layout. We present different relocation strategies for limited knowledge on the retrieval sequence. Experiments compare the performance of the different relocation strategies.

PART I

Straddle carrier allocation problem

Vessel and truck turnaround times determine the competitiveness of a container terminal. The time needed for transport and storage operations within the terminal highly influences turnaround times. Our objective is to use announced arrival times and announced volumes to optimize the allocation of internal handling equipment.

This study focuses on multimodal container terminals using only manned straddle carriers for internal transport and storage operations. In this case, straddle carriers are shared among different transport modes and the capacity of the terminal can be adapted via the number of drivers. We deal with the straddle carrier allocation problem at a tactical level. We present a model that allocates available straddle carriers to different transport modes with the objective to minimize overall delays at the terminal. We show how results can be used to determine the number of straddle carriers (drivers) needed for the next day. We also extend the study to evaluate the impact of a truck appointment system on overall delays at the terminal.

Chapter 2 introduces the straddle carrier allocation problem (SCAP) and different service strategies that may be applied for different transport modes. It summarizes literature on related problems. Chapter 3 represents SCAP as a network flow problem and presents a linear mixed integer program to solve it. First, a core model is implemented. This model is then extended to represent different service strategies. We discuss the complexity of the resulting models. Chapter 4 reports the results of a case study for the Grand Port Maritime de Marseille. It also evaluates the allocation proposed by the optimization model in a stochastic environment. Chapter 5 extends the model to a terminal using a truck appointment system and evaluates the impact of the truck appointment system on overall delays at the terminal.

Chapter 2

Straddle carrier allocation problem (SCAP)

This chapter introduces the straddle carrier allocation problem. It discusses different service strategies that may be applied for different transport nodes and introduces a notation to describe them. It also summarizes literature on resource allocation and task-scheduling problems at container terminals.

2.1 Problem description

The straddle carrier allocation problem (SCAP) occurs at container terminals serving several transport modes - like trucks, trains, barges and vessels - and using manned straddle carriers for all storage and transport operations. Terminals using manned straddle carriers can adapt their capacity from day to day via the number of hired dockers. We focus on container terminals where straddle carriers are allocated to one transport mode or one vehicle for a specified duration. This simplifies the work for drivers and allows reallocation of straddle carriers only at discrete points in time. The decision horizon of the straddle carrier allocation problem is one working day and includes two decisions at a tactical level: 1) How many straddle carriers are needed for the next day? 2) How to allocate these straddle carriers to the different transport modes and vehicles. The objective is to reduce delays at the terminal to reduce turnaround times and to increase the competitiveness of the terminal.

The straddle carrier allocation problem occurs when planning for the next day. New technologies make it possible to obtain reliable information on arrivals and volumes of vessels, trains and barges. But, the terminal has no or little direct information on future truck arrivals. However, we were able to obtain rather accurate forecasts on the number of arriving trucks and their distribution over the working day (Dauzère-Pérès et al.; 2012). Determining exact arrival times was out of scope of this problem. The forecast mechanism was developped within the ESPRIT project in cooperation with the system provider MGI[1] and the container terminal Seayard[2]. The work carried out in the context of the ESPRIT project is not detailed further in this document, since we did not implement a model combining the forecast mechanism with the straddle carrier allocation model. Instead, we assume that accurate forecasts on the arrival rate of trucks are available.

[1] http://www.gyptis.fr/
[2] http://www.seayard.com

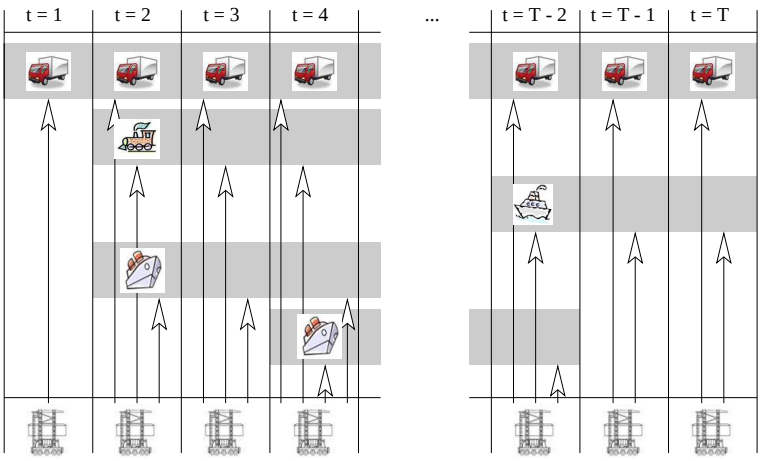


Figure 2.1: Straddle carrier allocation problem

We thus assume that arrivals and volumes of trains, barges and vessels are known and that the number of trucks arriving over the working day can be estimated. But, exact times when containers have to be picked up and exact travel times within the yard cannot be known. Detailed scheduling and routing decisions are not part of the problem. They are performed in real time by the Terminal Operating System (TOS) with the set of allocated straddle carriers. Major disruptions (e.g., the delayed arrival of a vessel) can be handled by solving the allocation problem with updated information.

The optimal allocation depends on the expected workload. We divide the working day into T time periods, as straddle carriers can only be reallocated at discrete points in time. The duration of each time period is defined by the terminal operator. We do not distinguish import and export containers as both require transport and storage operations by one straddle carrier. We use the term task to refer to all operations to be executed for a single container. The terminal has reliable information on the number of arriving vehicles I, on the arrival period r_i and on the number of tasks p_i per vehicle i. All tasks have to be executed at the end of the working day. The capacity of the terminal depends on the number of available straddle carriers s and on the average number of tasks a straddle carrier can execute per period h. Figure 2.1 illustrates the straddle carrier allocation problem for a terminal serving trucks, trains, barges and vessels.

Trucks, trains, barges and vessels are served with different service strategies. The chosen strategy depends on cargo volume, operating costs, knowledge and reliability of arrival and due dates and the used handling equipment. In the sequel, we use the term vehicle to refer to a single vehicle and the term transport mode to refer to all vehicles of the same transport mode. We introduce the $\alpha|\beta|\gamma$ notation, similar to the notation introduced by Graham et al. (1979) for scheduling problems, to describe the service strategy for a transport mode[3]. α specifies if straddle carriers are allocated to exactly one vehicle or to all vehicles of the same transport mode within one period. β specifies characteristics of the service strategy like vehicle arrival and due dates or limits on the maximum throughput per period due to the interaction with other equipment or space restrictions. γ represents different ways to measure delays. This may be the time a vehicle spends at the terminal, the number of

[3]We assume that all vehicles of the same transport mode are served with the same strategy. If this is not the case, the notation can also be used to describe the service strategy for a single vehicle.

Table 2.1: $\alpha|\beta|\gamma$ notation to describe the service strategy of one transport mode at a container terminal

1. Staddle carrier allocation (α)

ded	Straddle carriers are allocated to exactly one vehicle per period
shar	Straddle carriers may be shared among several vehicles of the same transport mode per period
shar+	Straddle carriers may be shared among vehicles of different transport modes per period

2. Service constraints (β)

r_v	Service starts after the arrival of the vehicle
d_v	Service has to be finished prior to the departure of the vehicle
$\tilde{d}_v$	Service should be finished prior to the preferred departure of the vehicle, but finishing later is possible at a cost
$p_v = 1$	Every vehicle requires exactly one container movement
non-incr	Once the service of a vehicle has started, the number of allocated straddle carriers may not increase
$\max_v$	Maximal throughput per vehicle per period is limited
$\max_m$	Maximal throughput per transport mode per period is limited

3. Penalty of delays (γ)

$\sum C_v$	Vehicles should be served as fast as possible - each period at the terminal is penalized
$\sum w_v C_v$	Like $\sum C_v$ but with different delay costs for vehicles
$\sum T_v$	Vehicles should be served prior to their due dates - each period at the terminal after its due date is penalized
$\sum w_v T_v$	Like $\sum T_v$ but with different delay costs for vehicles
$\sum U_c$	Container movement requests may remain unexecuted at a cost
$\sum S_v$	The number of shifts working on a vehicle should be minimized

non-executed tasks when the vehicle leaves the terminal or the number of shifts working on the vehicle. Table 2.1 presents values that may be taken by α, β and γ. Index v stands for vehicles, index m for transport modes and index c for container movement requests.

2.2 Related literature

This section summarizes literature on the resource allocation problem and the task-scheduling problem at container terminals. The resource allocation problem determines the number of resources to use for a specific set of tasks, whereas the task-scheduling problem assigns and schedules tasks on a predefined set of resources. For automated terminals, task assignment and scheduling predominant at an operational level as the capacity of AGVs cannot be modified on a daily basis and AGVs are shared between all vessels. Due to the focus on vessel turnaround times, most studies aim to minimize vessel service times.

Resource allocation

Only few studies tackle the problem of how to allocate internal handling equipment. To the best of our knowledge none of the existing literature has studied the straddle carrier allocation problem with the aim to minimize overall delays at the terminal.

Gambardella et al. (2001) and Kozan (2000) model the resource allocation problem as a network flow problem. Gambardella et al. (2001) determine a cost optimal allocation of quay cranes, transport vehicles and yard cranes that serves all vessels. They formulate the problem as a network design problem and present a mixed integer linear program, where containers to be moved are modeled as flows and arc capacities are limited by the number of allocated resources. The discrete-event simulation described in Gambardella et al. (1998) verifies the feasibility of the obtained solution. Kozan (2000) presents a network model to analyze container progress at the terminal. The aim is to minimize total throughput time of containers from their arrival to their departure (traveling and handling times). The model is meant to serve as a decision support system in the context of investment appraisal, but not for improvements on operational methods.

Kang et al. (2008) and Alessandri et al. (2008) apply queuing models to solve the resource allocation problem. Kang et al. (2008) present two models to optimize the size of the transportation fleet (cranes and trucks) for vessel unloading operations at container terminals. A cyclic queue model studies the steady-state port throughput which yields the optimum fleet size for long-term operations. A Markovian decision model determines optimal policies for fleet management in real time minimizing fleet operating costs and vessel waiting costs. Alessandri et al. (2008) present a predictive control approach to allocate available resources to minimize turnaround times. Waiting queues represent arrivals of vessels, trucks and trains, as well as containers to be loaded/unloaded and the presence of containers within the yard. The handling capacities for the various queues depend on the number of resources and their handling rates.

Vis et al. (2005) propose an integer linear program minimizing the number of vehicles required to transport containers between the quay and the yard. Quay crane and yard crane activities are represented by time-windows assigned to each container. Possible sequences of containers with regard to their time-windows are presented as directed paths. The minimum number of directed paths serving each container exactly once equals the minimum number of vehicles needed to execute the tasks. Evaluation by simulation shows that the analytical model performs well, but that it slightly underestimates the required fleet size.

Task-scheduling

Most studies schedule containers with the objective to minimize the vessel makespan; only few papers consider task-scheduling with the objective to minimize truck delays. Almost all studies assume to know exact container pick-up and delivery dates as well as precedence constraints among containers. Other studies evaluate different dispatching strategies.

Das and Spasovic (2003) develop an assignment algorithm that dynamically matches straddle carriers to waiting trucks. The objective is to minimize truck serving times and empty travel of straddle carriers. They show that their algorithm outperforms two basic dispatching approaches. They also evaluate how reducing the number of straddle carriers impacts the performance. Hartmann (2004) develops a general model to assign jobs to

container terminal resources and to temporally arrange the jobs with regard to precedence constraints and sequence-dependent resource setup times. He presents exemplary applications to schedule straddle carriers, automated guided vehicles, stacking cranes and workers who handle reefer containers. He introduces priority rule based heuristics and a genetic algorithm to solve these problems. The straddle carrier application includes container movements between quay, yard and landside gates. The objective is to serve quay cranes and inland transport modes at time and to reduce empty travel of straddle carriers. Lee et al. (2010) present a mixed integer program and two heuristics: a neighborhood search a method based on a genetic algorithm and on a minimum cost flow network model. The genetic algorithm represents the ready time of jobs and the minimum cost flow model decodes the chromosome to determine the job sequence. Their objective is to minimize service times. Skinner et al. (2013) present a mathematical model and a genetic algorithm where a chromosome represents a set of jobs and the number of jobs assigned to each straddle carrier. Their objective is to minimize travel and crane waiting times.

Böse et al. (2000) evaluate different strategies to dispatch straddle carriers to gantry cranes; they compare vessel turnaround times and straddle carrier utilization rates. Results show that dynamic strategies, where straddle carriers are not assigned to a single quay crane, but shared among several vessels, improve the performance of the unloading/loading process. Further improvements are obtained by applying an evolutionary algorithm. Bish et al. (2005) present greedy-algorithms to come up with dispatching strategies for discharging a single vessel based on container precedence constraints. They identify the absolute and asymptotic worst-case performance ratios for these heuristics. Nguyen and Kim (2009) present a mixed integer programming model to assign delivery tasks for vessels to automated lifting vehicles. Buffer constraints at the quay are converted into time window constraints. They present a heuristic algorithm for this model and to compare different dispatching strategies. Nguyen and Kim (2010) propose different dispatching strategies taking into account the uncertainty of vehicle travel times. They propose a mixed integer programming model using sequence lists for tasks and penalties for delayed delivery and a heuristic dispatching algorithm. The model and the heuristic are based on the work of Kim and Bae (2004) for the dispatching of vehicles with deterministic travel times. A simulation study proved that the heuristic with uncertain travel times outperforms the heuristic with deterministic travel times with regard to delay of quay cranes and moves per vehicle, but at the expense of greater computational time.

Briskorn et al. (2006) present an inventory based approach for dispatching vehicles. They consider quay cranes as customers and AGVs as goods. The inventory level of a quay crane is the number of AGVs in the buffer. The inventory level should not be zero and shouldn't be too high. This formulation does not depend on highly unreliable estimations of driving times, completion times, due dates and tardiness. Simulation shows that the inventory based model leads to better productivity than the scheduling based formulation. Froyland et al. (2008) deal with the operating of a landside exchange area that is served by multiple semi-automated rail mounted gantry cranes. The problem is divided into three sequential subproblems. First, all container moves are assigned to an hourly level by an integer program. Then, import containers are assigned to positions in the gantry-straddle carrier interface via integer programs. Finally, containers are assigned to short-term stacking positions and precise RMG operations are planned within each hour by an online algorithm.

Kim and Kim (1999) formulate a integer programming model to route a single straddle carrier during the loading operation of export containers on a vessel. They determine the number of containers picked up at each area in the yard (yard-bay), as well as the sequence of yard-bays visited during the tour. They apply dynamic programming to obtain a route minimizing the total travel distance of the straddle carrier. Balev et al. (2009) formulate the straddle carrier routing problem as a dynamic vehicle routing problem with pick-up and delivery and time windows. Their objectives are to minimize the number and the traveled distance of straddle carriers, as well as the delay of tasks. Discrete-event simulation compared different scheduling policies.

Integrated scheduling

Recent studies focus on the integrated scheduling of different handling equipment. They integrate quay cranes, AGVs or yard trucks and yard cranes to minimize the vessel makespan. Meersmans and Wagelmans (2001) present a beam search heuristic and several dispatching rules for vessel loading operations. Chen et al. (2007) present a hybrid flow shop scheduling model for vessel loading and unloading operations. Lau and Zhao (2008) present a mixed integer programming model for loading and unloading operations. They apply a multi-layer genetic algorithm as well as a genetic algorithm plus maximum matching to this problem. Cao et al. (2010) present a mixed integer programming model for vessel loading operations and two methods based on Bender's decomposition to solve it. Zeng and Yang (2009) propose an integrated optimization simulation approach with a genetic algorithm to improve the given container sequence and a simulation model to evaluate a given schedule. Chen, Langevin and Lu (2013) formulate the problem as a constraint programming-model. They develop an iterative three-stage algorithm: generating crane schedules, solving the multiple-truck routing problem and constructing a complete solution.

Other studies deal with the scheduling of equipment and the storage allocation in an integrated way. The objective is to minimize container transfer times and service times. Kozan and Preston (2006) apply an iterative search algorithm that integrates a container transfer model with a container location model in a cyclic fashion. A genetic algorithm, a tabu search and a tabu search/genetic algorithm hybrid are used to solve the problem. Hadjiconstantinou and Ma (2009) mention an integer programming model minimizing handling and storage costs. A simulation model based on the object-oriented paradigm validates the results and proves the robustness of the optimization model. Lee et al. (2009) present a mixed integer program and solve the problem with an hybrid insertion algorithm. Wu et al. (2013) formulate a linear mixed integer program and a non-linear mixed integer program. They solve the problem with a genetic algorithm where a chromosome represents the vehicle and the yard location associated to each container.

Human resources

The allocation of manned equipment is highly related with human resource management. Legato and Monaco (2003) solve the manpower planning problem with special focus on the uncertainty of workforce demand. They consider that the arrival time of a container vessel is known just a few hours in advance. They propose a two phase solution. First a monthly planning constructs a set of feasible working schedules. This planning includes some flexibility on the base of union agreements. Then, a daily planning assigns each worker to a shift, a gang and a task. Fancello et al. (2011) continued this work by proposing

a dynamic learning predictive algorithm based on neural networks to predict ship delays. They also propose an optimization algorithm for the daily allocation problem. Kim et al. (2004) search feasible solutions for the scheduling of operators of handling equipment. The major constraints include: restrictions on the minimum workforce assignment to each time slot, the maximum total operating time per operator per shift, the minimum and maximum consecutive operating times for an operator, types of equipment that can be assigned to each operator and the available time slots for each operator or piece of equipment. The problem is defined as a constraint-satisfaction problem.

Chapter 3

Mixed integer program for SCAP

Our objective is to provide the terminal operator with a tool to 1) estimate the number of straddle carriers needed for the next day to handle the expected workload and 2) to propose a possible allocation of these straddle carriers to different trucks, trains, barges and vessels to minimize overall delays at the terminal.

We model the straddle carrier allocation problem as a network flow problem where containers to be moved are flows and arc capacities are limited by the capacity of allocated straddle carriers. Section 3.1 introduces the core model allocating straddle carriers to different vehicles to serve each vehicle within its imposed time window. Section 3.2 presents extensions to the core model to represent different service strategies. Section 3.3 evaluates the impact of different model formulations and input parameters on the run time. Section 3.4 briefly discusses the complexity of the arising problems. Section 3.5 presents an aggregated network flow model and compares its performance to the core model. Section 3.6 summarizes the chapter.

3.1 Vehicle network flow model (core model)

We model the straddle carrier allocation problem as a network flow problem. This implementation is inspired by the model in Gambardella et al. (2001). But, our objectives and our formulation differ from theirs. Their objective is to find a cost minimal allocation to serve each vessel within its time window. Whereas we model the entire terminal with special focus on landside transportation modes and aim to minimize overall delays at the container terminal.

Figure 3.1 illustrates our modeling approach. It represents a terminal with two vehicles arriving over the day. Each network flow model represents one vehicle - thus, called vehicle model. The round nodes stand for the discrete time periods of the working day. The rectangular nodes, which are sources of flow, represent the arrival of vehicles with their associated demand for container movement requests p_1^m and p_2^m. Each source is connected to the period corresponding to the arrival of the vehicle: period 1 for vehicle 1 and period 4 for vehicle 2. The square nodes represent sinks of flow. Flows $W_{i,t}^m$ from a period node to a sink represent the number of container movements executed per vehicle per period. Arc capacities limit the maximum number of container movements that may be executed by the capacity of allocated straddle carriers. Flows $Z_{i,t}^m$ between two periods represent unexecuted tasks which are transferred to the next period. All arcs delaying tasks after the due date of

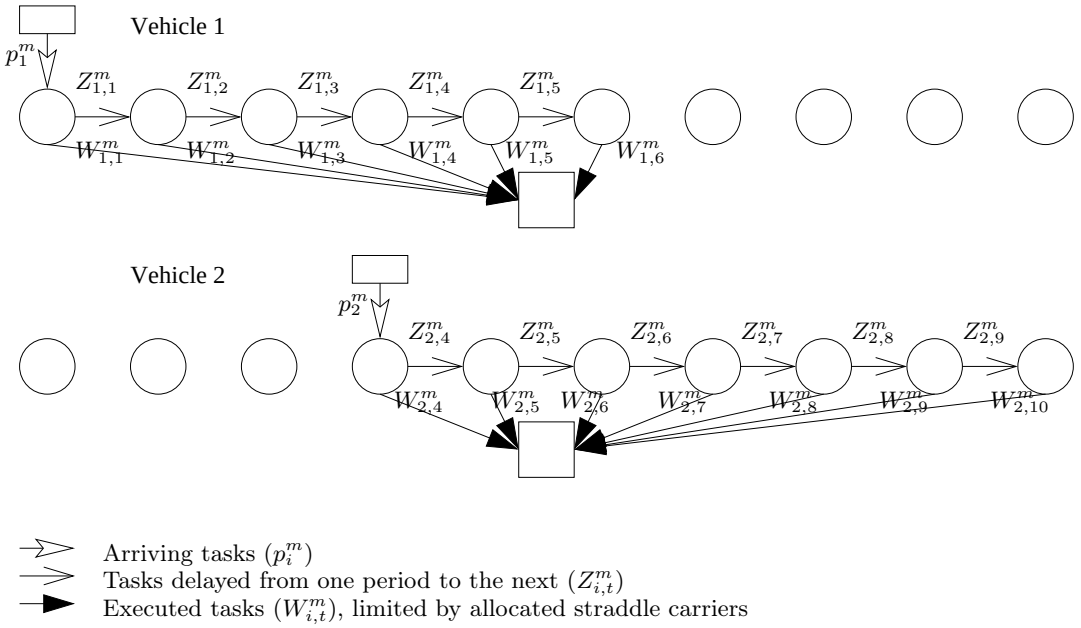


Figure 3.1: Scheme of the vehicle network flow model

the vehicle have a capacity of 0; here, arcs after period 6 for vehicle 1 and after period 10 for vehicle 2. Note that the different vehicle models are independent despite the fact that they have to share the available straddle carriers. We use this modularity later on to include vehicle specific constraints to each submodel.

We now state the mixed integer linear program formulating these network flows. In our case, the expected workload and the capacity of the terminal represent the situation at a container terminal. The expected workload is determined by the number of vehicles with their arrival times and volumes. The capacity is given by the number of available straddle carriers and their average handling capacity. We use the following parameters to describe the expected workload and the capacity of the terminal:

T	Number of time periods describing the working day
M	Number of transport modes being served at the terminal
I^m	Number of vehicles of transport mode m arriving at the terminal
t	Index of a time period, $t = 1, \ldots, T$
m	Index of a transport mode, $m = 1, \ldots, M$
i	Index of a vehicle of transport mode m, $i = 1, \ldots, I^m$
r_i^m	Period t in which vehicle i of transport mode m arrives at the terminal
d_i^m	Period t in which vehicle i of transport mode m has to be ready for departure ($d_i^m \leq T$)
p_i^m	Total number of tasks to be carried out for vehicle i of transport mode m
s_t	Number of straddle carriers available at period t
h^m	Average number of tasks a straddle carrier serving transport mode m can handle per period, we assume that $h^m \in \mathbb{N}^+$ and that $h^m \geq 1$ $\forall m = 1, \ldots, M$

Variables indicate the number of straddle carriers to allocated to each vehicle per period and the number of delayed and executed tasks resulting from this allocation.

$X_{i,t}^m$	Number of straddle carriers allocated to vehicle i of transport mode m in period t
$W_{i,t}^m$	Number of tasks executed for vehicle i of transport mode m in period t
$Z_{i,t}^m$	Number of non-executed tasks for vehicle i of transport mode m which are transferred from period t to period $t+1$

These parameters and variables enable us to formulate the straddle carrier allocation problem. Straddle carriers are not shared and each vehicle has to be served within its specified time window. We call this model the core model.

$$W_{i,t}^m \leq h^m \cdot X_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = 1,\ldots,T \tag{3.1}$$

$$Z_{i,t}^m = p_i^m - W_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = r_i^m \tag{3.2}$$

$$Z_{i,t}^m = Z_{i,t-1}^m - W_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = r_i^m+1,\ldots,T \tag{3.3}$$

$$Z_{i,d_i^m}^m = 0, \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m \tag{3.4}$$

$$\sum_{m=1}^{M}\sum_{i=1}^{I^m} X_{i,t}^m \leq s_t \qquad \forall t = 1,\ldots,T \tag{3.5}$$

$$X_{i,t}^m \in \left\{\mathbb{N}^+, \mathbb{R}^+\right\} \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = 1,\ldots,T \tag{3.6}$$

$$W_{i,t}^m, Z_{i,t}^m \in \mathbb{R}^+ \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = 1,\ldots,T \tag{3.7}$$

Constraint (3.1) ensures that the number of executed tasks per vehicle per period does not exceed the capacity of straddle carriers allocated to this vehicle. Constraints (3.2) and (3.3) formulate the mass balance constraints for arriving, executed and delayed tasks for each vehicle. They also make sure that no tasks are executed prior to a vehicle's arrival. Constraint (3.4) imposes that each vehicle is completely served prior to its deadline. Constraint (3.5) guarantees that the number of allocated straddle carriers does not exceed the number of straddle carriers available at the terminal. Constraint (3.6) indicates that $X_{i,t}^m$ may take real or natural numbers. The choice depends on the possibility to share straddle carriers among different vehicles and is presented in detail in section 3.2. Tasks have to be executed completely within one period (the container is transported from its origin to its destination or not at all). This implies that $W_{i,t}^m$ and $Z_{i,t}^m$ have to take integer values. For $h^m \in \mathbb{N}^+$ together with Constraints (3.1) to (3.3), $W_{i,t}^m$ and $Z_{i,t}^m$ will always take integer values even if defined as continuous variables like in Constraint (3.7). In Section 3.3.2, we see that using continuous variables decreases solution times.

Constraints (3.8) to (3.10) are optional and have no impact on the objective function. They prevent allocating excess straddle carriers and render the solution more comprehensible. Constraint (3.8) imposes that each allocated straddle carrier executes at least one task. Constraints (3.9) and (3.10) make sure that no straddle carrier is allocated to a vehicle prior to its arrival or after its departure. In Section 3.3.2, we show that these constraints have small, rather positive impacts on solution times.

$$W_{i,t}^m \geq h^m \cdot (X_{i,t}^m - 1) + 1 \qquad \forall m = 1, \ldots, M, i = 1, \ldots, I^m, t = 1, \ldots, T \tag{3.8}$$

$$X_{i,t}^m = 0 \qquad \forall m = 1, \ldots, M, i = 1, \ldots, I^m, t = 1, \ldots, r^m - 1 \tag{3.9}$$

$$X_{i,t}^m = 0, \qquad \forall m = 1, \ldots, M, i = 1, \ldots, I^m, t = d_i^m + 1, \ldots, T \tag{3.10}$$

3.2 Extensions for different service strategies

This section illustrates how the core model may be adapted to the different service strategies presented in Section 2.1. Let $\{1, \ldots, m'\}$ be the transport modes for which the specified service strategy applies.

3.2.1 Straddle carrier allocation

Straddle carriers may be allocated to one vehicle, shared among vehicles of the same transport mode or shared between all vehicles. Manned equipment is often dedicated to one vehicle or a group of vehicles for a given time period for organizational reasons.

ded$|\beta|\gamma$ If straddle carriers are dedicated to one vehicle, excess capacity of one straddle carrier in one period may not be used to serve other vehicles. Constraint (3.11) allocates a straddle carrier to exactly one vehicle per period.

$$X_{i,t}^m \in \mathbb{N}^+ \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, t = 1, \ldots, T \tag{3.11}$$

shar$|\beta|\gamma$ Straddle carriers are shared among all vehicles of the same transport mode within one period; but not between transport modes. We introduce a new variable to model this allocation strategy:

X_t^m Total number of straddle carriers allocated to all vehicles i of transport mode m in period t

Constraint (3.12) allows partial allocation and thus sharing of straddle carriers. Constraints (3.13) and (3.14) prevent sharing among vehicles of different transport modes. Constraint (3.15) limits the total number of allocated straddle carriers. It is modified version of Constraint (3.5) taking into account shared and dedicated straddle carriers.

$$X_{i,t}^m \in \mathbb{R}^+ \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, t = 1, \ldots, T \tag{3.12}$$

$$\sum_{i=1}^{I^m} X_{i,t}^m \leq X_t^m \qquad \forall m = 1, \ldots, m', t = 1, \ldots, T \tag{3.13}$$

$$X_t^m \in \mathbb{N}^+ \qquad \forall m = 1, \ldots, m', t = 1, \ldots, T \tag{3.14}$$

$$\sum_{m=m'+1}^{M} \sum_{i=1}^{I^m} X_{i,t}^m + \sum_{m=1}^{m'} X_t^m \leq s_t \qquad \forall t = 1, \ldots, T \tag{3.15}$$

shar$+|\beta|\gamma$ If straddle carriers are shared among all vehicles of all transport modes, the scheduling of tasks becomes more important. Our model can be adapted to this case via Constraint (3.16), but shouldn't be applied since it does not include scheduling decisions.

$$X_{i,t}^m \in \mathbb{R}^+ \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, t = 1, \ldots, T \tag{3.16}$$

3.2.2 Service constraints

Service constraints represent arrival and due dates of vehicles, as well as space or capacity restrictions.

$\alpha|r_v|\gamma$ No tasks may be executed prior to the arrival of a vehicle. This specification is valid for almost every vehicle and is already included in the core model and in some of the constraints presented below. The model may be adapted easily to the few exceptions where containers may be prepared in advance by setting $r_v = 1$.

$\alpha|d_v|\gamma$ No tasks may be executed after the due date of a vehicle. This constraint holds for all vehicles and is already included in the core model. The due date is set to the imposed departure time of the vehicle or to the end of the working day[1].

$\alpha|\max_v|\gamma$ The maximum number of tasks that can be executed per vehicle per period is limited (e.g. linkage with other equipment or space restrictions). The parameter q_i^m represents this limit.

$$W_{i,t}^m \leq q_i^m \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, t = 1, \ldots, T \tag{3.17}$$

$\alpha|\max_m|\gamma$ The maximum number of tasks that can be executed per transport mode per period is limited (e.g. space restrictions). The parameter q^m represents this limit.

$$\sum_{i=1}^{I^m} W_{i,t}^m \leq q^m \qquad \forall m = 1, \ldots, m', t = 1, \ldots, T \tag{3.18}$$

$\alpha|p_v = 1|\gamma$ Every vehicle requires exactly one container movement. This property does not change the core model. But in some cases it may be used to simplify the modeling of the objective function.

$\alpha|r_v$,**non-incr** $|\gamma$ We impose a non-increasing number of straddle carriers serving the vehicle. We allow withdrawing superfluous straddle carriers, but they cannot be reallocated to the vehicle later on. We introduce a new variable, $D_{i,t}^m$, which indicates if the service of a vehicle has already been started in $t-1$. This variable is defined by Constraints (3.19) and (3.20). Constraint (3.21) imposes a non-increasing number of allocated straddle carriers per vehicle once the service of a vehicle has started.

[1]Vessels spending several days at the terminal are split into several vessels: one for each day. Their arrival times, due dates and volumes are adapted.

$$D_{i,t}^m = \begin{cases} 0 & \text{if service on vehicle } i \text{ has not started prior to period } t \text{ } (t \text{ excluded}), \\ 1 & \text{otherwise.} \end{cases}$$

$$D_{i,t}^m \in \{0,1\} \qquad \forall m = 1,\dots,m', i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \quad (3.19)$$

$$D_{i,t}^m \geq \frac{p_i^m - Z_{i,t-1}^m}{p_i^m} \qquad \forall m = 1,\dots,m', i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \quad (3.20)$$

$$X_{i,t}^m \leq X_{i,t-1}^m - s_t \cdot (D_{i,t}^m - 1) \quad \forall m = 1,\dots,m', i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \quad (3.21)$$

3.2.3 Penalty of delays

We either penalize the completion time of a vehicle, its delayed departure or the number of non-executed tasks.

$\alpha|r_v, d_v, \beta|\sum C_v$ We want to minimize the time a vehicle spends at the terminal (waiting and service times) and penalize each period it spends at the terminal. A binary variable, $Y_{i,t}^m$, indicates if the vehicle is completely served and may leave the terminal or not. Constraints (3.22) and (3.23) together with the objective function assert that $Y_{i,t}^m$ equals zero if and only if the service of a vehicle is finished.

$$Y_{i,t}^m = \begin{cases} 0 & \text{if the service of vehicle } i \text{ of transport mode } m \text{ is finished in period } t, \\ 1 & \text{otherwise.} \end{cases}$$

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} \sum_{t=r_i^m}^{T} Y_{i,t}^m$$

$$Y_{i,t}^m \geq \frac{Z_{i,t}^m}{p_i^m} \qquad \forall m = 1,\dots,m', i = 1,\dots,I^m, t = r_i^m,\dots,T \quad (3.22)$$

$$Y_{i,t}^m \in \{0,1\} \qquad \forall m = 1,\dots,m', i = 1,\dots,I^m, t = r_i^m,\dots,T \quad (3.23)$$

$\alpha|r_v, d_v, \beta|\sum w_v C_v$ Some vehicles are considered more important than others and should be served faster. We introduce weights w_i^m to represent these priorities. Like before, we add Constraints (3.22) and (3.23) to determine the time spent at the terminal.

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} \sum_{t=r_i^m}^{T} w_i^m \cdot Y_{i,t}^m$$

$\alpha|r_v, d_v, p_v = 1, \beta|\sum C_v$ We penalize each delayed tasks to minimize the time vehicles spend at the terminal. Since every vehicle requires exactly one container movement, each delayed task is equivalent to a delayed vehicle. No variables or constraints have to be added.

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} \sum_{t=r_i^m}^{T} Z_{i,t}^m$$

$\alpha|r_v, \tilde{d}_v, d_v, \beta|\sum T_v$ The vehicle has a preferred finishing time $\tilde{d}_v$ with $\tilde{d}_v < d_v$. The objective is to serve the vehicle before its preferred finishing time. Each period it spends at the terminal after period $\tilde{d}_v$ is penalized in the objective function. We add Constraints (3.22) and (3.23) but penalize only delays occurring after period $\tilde{d}_i^m$.

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} \sum_{t=\tilde{d}_i^m}^{T} Y_{i,t}^m$$

It is possible to include weights w_i^m into the objective function to express priorities among different vehicles. For $p_v = 1$, we can replace $Y_{i,t}^m$ by $Z_{i,t}^m$ and do not need Constraints (3.22) and (3.23).

$\alpha|r_v, d_v, \beta|\sum U_c$ Tasks should be executed prior to the departure of the vehicle. But, they may remain unexecuted at a cost. Variable U_i^m and Constraints (3.24) and (3.25) determine the number of non-executed tasks.

U_i^m Number of tasks which are not executed for vehicle i of transport mode m at its departure

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} U_i^m$$

$$Z_{i,d_i^m}^m - U_i^m = 0 \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m \tag{3.24}$$

$$U_i^m \in \mathbb{R}^+ \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m \tag{3.25}$$

$\alpha|r_v, d_v, \beta|$- Delays are forbidden and each vehicle has to be served prior to its due date. To verify if such an allocation exists we solve the problem $\alpha|r_v, d_v, \beta|\sum U_c$. Either an allocation where no delays occur exists or the problem is infeasible.

$\alpha|r_v, d_v, \beta|\sum S_v$ Container terminals divide the working day into several shifts. We assume that shifts have fix start and end times. The objective is to serve each vehicle with the fewest number of shifts. Parameters J, b_j and e_j are introduced to represent the repartition of the working day into shifts. Variables $H_{i,j}^m$ indicate shifts that work on the vehicle. Constraints (3.27) to (3.28) determine this variable. Each shift working on the vehicle is penalized in the objective function.

$$H_{i,j}^m = \begin{cases} 1 & \text{if vehicle } i \text{ of transport mode } m \text{ is served during shift } j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.26}$$

$$\min \sum_{m=1}^{m'} \sum_{i=1}^{I^m} \sum_{j=1}^{J} H_{i,j}^m$$

$$H_{i,j}^m \cdot s_t \geq X_{i,t}^m \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, j = 1, \ldots, J, t = b_j, \ldots, e_j \tag{3.27}$$

$$H_{i,j}^m \in \{0, 1\} \qquad \forall m = 1, \ldots, m', i = 1, \ldots, I^m, j = 1, \ldots, J \tag{3.28}$$

3.3 Sensitivity analysis

We carry out a sensitivity analysis to evaluate how the model formulation and the input parameters impact the solution time. We describe the generated instances. We compare different variable domains and analyze the impacts of the non-mandatory constraints (3.8) to (3.10). We also analyze the impact of different input parameters on the solution time. We vary the number of vehicles arriving at the terminal and their volume, the number of time intervals representing the time horizon, the number of available straddle carriers, the length of time windows and the possibility to share straddle carriers.

We observe that the non-mandatory constraints enable us to solve slightly more problems within the given time limit. This proves that despite increasing the clarity of the solution, these constraints have also positive effects on the solution process. Results also show that input parameters are highly interacting and that it is not possible to determine easy rules to predict the influence of these parameters on the solution time.

3.3.1 Instance generation

We consider a small fictitious container terminal - inspired by a terminal at the Grand Port Maritime de Marseille - for the sensitivity analysis. The terminal serves four different transport modes ($M = 4$) with different service strategies and straddle carriers are not shared among different transport modes. The working day is divided into 14 periods ($T = 14$) and each straddle carrier may execute six ($h = 6$) container movements per period.

For transport modes 1 to 3, few vehicles with high volumes (e.g., vessels, barges and trains) arrive over the time horizon. Straddle carriers are dedicated to single vehicles. We set $I^1 = I^2 = I^3 = 1$ and $p_1^1 = p_1^2 = p_1^3 = 100$. Transport mode 1 is served with strategy $\text{ded}|r_v, d_v|\sum C_v$ - the vehicle has to be served within the prescribed time window and each period it stays at the terminal is penalized; transport mode 2 is served with strategy $\text{ded}|r_v, d_v|-$ - the vehicle has to be served within the prescribed time window; transport mode 3 is served with strategy $\text{ded}|r_v, d_v|\sum U_c$ - the vehicle leaves the terminal at a fixed time and non-executed task are penalized. Vehicles of transport modes 1, 2 and 3 stay 7 periods at the terminal. Arrival times (r_1^1, r_1^2, r_1^3) are set to a random number between 1 and $T - 7$ and departure times (d_1^1, d_1^2, d_1^3) are set to $r_i^m + 7$.

For transport mode 4, a great number of vehicles with only one container movement request (e.g. trucks) arrive over the time horizon. Straddle carriers are shared among different vehicles. We set $I^4 = 200$ and $p_i^4 = 1$. Transport mode 4 is served with strategy $\text{shar}|r_v, d_v, p_v = 1|\sum C_v$ - delaying a vehicle from one period to the next is penalized. All

Table 3.1: Experimental plan for the sensitivity analysis describing scenarios A to O with their input parameters

	I^1	I^2	I^3	r_i^1, r_i^2, r_i^3	d_i^1, d_i^2, d_i^3	p_i^1, p_i^2, p_i^3	I^4	r_i^4	d_i^4	p_i^4	T	h	$s(p,T,h)$
A	1	1	1	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	6
B	**3**	**0**	**0**	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**7**
C	**0**	**3**	**0**	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**7**
D	**0**	**0**	**3**	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**7**
E	**2**	**2**	**2**	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**10**
F	1	1	1	r_i^m	$r_i^m + 7$	**200**	200	r_i^m	T	1	14	6	**10**
G	1	1	1	r_i^m	$r_i^m + 7$	100	**100**	r_i^m	T	1	14	6	**5**
H	1	1	1	r_i^m	$r_i^m + 7$	100	**400**	r_i^m	T	1	14	6	**9**
I	1	1	1	r_i^m	$r_i^m + 7$	100	**800**	r_i^m	T	1	14	6	**13**
J	1	1	1	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**5**
K	1	1	1	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**7**
L	1	1	1	r_i^m	$r_i^m + 7$	100	200	r_i^m	T	1	14	6	**8**
M	1	1	1	$2 \cdot r_i^m$	$r_i^m + 14$	100	200	$2 \cdot r_i^m$	T	1	**28**	**3**	6
N	1	1	1	$r_i^m + 1$	$r_i^m + 5$	100	200	$r_i^m + 2$	T	1	14	6	6
O	1	1	1	$r_i^m - 1$	$r_i^m + 9$	100	200	$r_i^m - 2$	T	1	14	6	6

vehicles have to be served at the end of the day ($d_i^4 = T$). Arrival times (r_i^4) are set to a random number between 1 and T.

The number of available straddle carriers per period ($s_t = s$) is determined as a function of the total container movement requests, the number of time periods and the number of tasks a straddle carrier may handle per period. We set $s = \left\lceil \sum_{m=1}^{M} \sum_{i=1}^{I^m} p_i^m / (T \cdot h) \right\rceil$. We generate ten different instances (I to X) according to these specifications. In order to analyze the impact of different parameters on the solution time, we created scenarios (B to O) of the initial scenario A. Table 3.1 summarizes input parameters of scenarios A to O.

Scenarios B, C and D represent a terminal with only two transport modes with three vehicles of either transport mode 1, 2 or 3 and an unchanged transport mode 4. These scenarios allow us to analyze the impact of the chosen service strategy on the solution time. For transport modes 1 and 2, all tasks have to be executed prior to the departure of the vehicle. With 6 available straddle carriers not all tasks can be executed in time and scenarios B and C are infeasible. Therefore, we increase the number of straddle carriers to 7. Scenario D is feasible with 6 straddle carriers. But to compare its results with scenarios B and C, the number of straddle carriers is also increased to 7.

Scenarios E to I vary the number of tasks to be executed. Either the number of vehicles per transport mode is increased (E, H and I) or decreased (G) or the number of tasks per vehicle is increased (F). The number of available straddle carriers is adapted to the new

volumes by reapplying the formula presented above. For scenarios J to L the number of available straddle carriers is decreased/increased for an identical demand. This allows us to analyze the impact of changes in demand and available capacity.

For scenario M, the number of time periods is doubled. Since the time horizon is fixed to one working day, time periods become shorter. Doubling the number of time periods is identical to divide the length of each time period by two. We adapt the number of tasks a straddle carrier can handle per period ($h^m = 3$). Vehicle arrival times are set to $r'^m_i = 2 \cdot r^m_i$ or $r'^m_i = 2 \cdot r^m_i - 1$. Departure times for transport modes 1 to 3 are set to $d'^m_i = r'^m_i + 14$. Departure times for transport mode 4 remain unchanged at T. These scenarios enable us to analyze the impact of the number of time periods.

Scenarios N and O vary the length of the time window. In scenario N, the time window length is decreased from 7 to 5 by setting $r'^m_i = r^m_i + 1$ and $d'^m_i = r'^m_i + 5$. In scenario O, the time window length is increased from 7 to 9 by setting $r'^m_i = r^m_i - 1$ and $d'^m_i = r'^m_i + 9$. If $r'^m_i + 9 > T$, d'^m_i is set to T and r'^m_i to T - 9. These instances allow to analyze the impacts of the time vehicles may stay at the terminal on the solution time.

3.3.2 Impact of the model formulation

The core model presented in Section 3.1 contains constraints necessary to represent the straddle carrier allocation problem correctly (Constraints (3.1) to (3.7)) and additional constraints rendering the solution more comprehensible (Constraints (3.8) to (3.10)). We also discussed that variables $W^m_{i,t}$ and $Z^m_{i,t}$ may be integer of continuous. We compare four different models with different variable domains and with and without the non-mandatory constraints. We perceive that instances are solved faster if variables $W^m_{i,t}$ and $Z^m_{i,t}$ are defined as continuous rather than as integer variables. We also see that the non-mandatory constraints enable us to solve slightly more problems within the given time limit.

Constraints (3.8) to (3.10) reduce the solution space since they prevent the allocation of superfluous straddle carriers before, during and after the service of a vehicle. We suppose that these constraints have similar effects on the solution time, as all three limit the solution space. We compare four different model formulations:

- **Model M1:** $W^m_{i,t}, Z^m_{i,t} \in \mathbb{N}^+$, with Constraints (3.8) to (3.10)
- **Model M2:** $W^m_{i,t}, Z^m_{i,t} \in \mathbb{R}^+$, with Constraints (3.8) to (3.10)
- **Model M3:** $W^m_{i,t}, Z^m_{i,t} \in \mathbb{N}^+$, without Constraints (3.8) to (3.10)
- **Model M4:** $W^m_{i,t}, Z^m_{i,t} \in \mathbb{R}^+$, without Constraints (3.8) to (3.10)

We solve the four models using the commercial solver IBM ILOG CPLEX 12 on our instances I to X for scenarios A to O and compare the results. We limit the execution time to 600 seconds for each problem and measure the CPU time needed to solve the problem. If the solution process is not finished after the imposed time limit, we register the gap between the best found solution and the lower bound (relaxed solution).

Table 3.2 presents the results of these experiments. For each scenario, it indicates the number of feasible instances (out of 10) and the number of instances that were solved within

Table 3.2: Performance comparison of models M1 to M4 on scenarios A to O

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	avg.
feasible	10	10	10	10	10	10	10	10	10	7	10	10	10	8	10	9.7
Number of instances solved within 600 seconds																
M1	8	7	6	8	1	1	7	3	6	7	10	10	1	8	4	5.8
M2	10	10	8	10	3	10	10	10	10	7	10	10	3	8	9	8.5
M3	8	6	6	9	1	1	6	4	6	7	9	10	1	8	4	5.7
M4	10	9	7	10	3	9	10	10	10	7	10	10	2	8	10	8.3
Average CPU time in seconds for solved instances																
M1	172.2	59.7	0.3	0.8	0.2	262.6	164.7	50.6	183.8	76.2	70.1	0.2	184.1	3.9	147.4	91.8
M2	13.7	30.4	10.2	2.0	147.0	4.2	18.2	7.1	10.0	5.8	6.5	0.2	23.5	0.4	139.0	27.9
M3	149.8	0.4	1.2	51.4	0.2	344.5	139.9	118.2	163.8	67.1	26.4	0.2	130.2	3.2	111.5	87.2
M4	8.5	0.8	15.4	14.8	56.5	8.6	13.8	7.3	30.3	0.6	0.6	0.1	12.5	0.4	53.8	14.9
Average gap in percentage for unsolved instances																
M1	4.2	18.5	33.0	12.1	13.3	8.7	6.2	8.3	3.0	0.0	0.0	0.0	4.2	0.0	7.4	-
M2	0.0	0.0	21.7	11.0	9.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	9.1	-
M3	3.5	15.3	37.2	13.0	14.1	8.2	5.9	12.7	3.0	0.0	2.4	0.0	5.4	0.0	7.7	-
M4	0.0	3.4	17.9	8.6	10.0	8.1	0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.0	0.0	-

600 seconds for each model. It also shows average solution times for solved instances for each scenario and each model. For unsolved instances, it reports average gaps between the current solution and the lower bound for each scenario and model[2]. When comparing the average solution time and the average gap, we have to keep in mind that those averages are computed on the number of solved and unsolved instances which are not always identical for the models.

By comparing the results from model M1 with M3 and from model M2 with M4, we observe that it is generally advantageous to define $W^m_{i,t}, Z^m_{i,t}$ in $\mathbb{R}^+$ rather than in $\mathbb{N}^+$. For all scenarios A to O, more or as many instances are solved for models with real variables (M2 and M4) than for models with integer variables (M1 and M3). The difference is particularly large for scenario F. By using real variables, the average solution time may be decreased by more than 60%. Only for scenarios C and E the solution time of model M1 (respectively M3) is inferior to the solution time of model M2 (respectively M4). But, for these scenarios, M1 and M3 solve fewer instances than models M2 and M4.

To analyze the impacts of our non-mandatory Constraints (3.8) to (3.10) we compare the results of model M1 with M3 and the results from model M2 with M4. We observe that in average M1 (respectively M2) solves slightly more instance per scenario than M3 (respectively M4). But there is no dominance as M1 solves more instance than M3 for scenarios B,G and K and M3 solves more instances for scenarios D and H. M2 solves more instances than M4 for scenarios B, C, F and M and M4 solves more instances for scenario O. In average M3 (respectively M4) is solved 14 seconds faster than M1 (respectively M2). But, these values do not take into account that more instances are solved for models M1 and M3 than for models M2 and M4. This shows that those non-mandatory constraints may be included in the model to render the obtained solution more comprehensible and that they have rather positive impacts on the solution process.

We may conclude that model M2 performs at least as well as the other models and use it for further analysis; all reported results from now on were obtained with model M2.

3.3.3 Impact of input parameters

We analyze how different input parameters influence the solution time. We compare scenarios B to O with the basic scenario A to analyze the impacts of the chosen service strategies, the volumes to be handled, the number of available straddle carriers, the number of time periods and the length of time windows. We add scenarios B' to E' to analyze the impact of sharing straddle carriers on the solution time. Instances B' to E' are identical to instances B to E but straddle carriers may be shared between vehicles of the same transport mode.

Table 3.3 presents the solution time for each instance and each scenario. Infeasible instances are labeled with "inf". Instances that couldn't be solved because a lack of memory are labeled with "mem". An entry above 600 signifies that the time limit of 600 seconds was reached. We observe that the optimal solution is either found very quickly (<10 seconds) or that optimality cannot be shown within the given time limit. In the following, we discuss these results and the impact of input parameters in more detail.

[2]For scenario M, several instances could not be solved because of a lack of memory. The average considers only instances with sufficient memory space.

Table 3.3: Average CPU time in seconds for each instance for scenarios A to O with a time limit of 600 seconds

	I	II	III	IV	V	VI	VII	VIII	IX	X	avg.
A	0.5	0.8	1.0	0.9	1.0	0.3	70.5	0.9	60.2	0.8	13.7
B	0.3	2.1	0.3	0.3	1.2	0.3	0.3	0.3	297.7	1.6	30.4
C	1.3	600.1	6.0	0.4	22.2	0.1	0.1	0.1	600.1	51.7	128.2
D	0.7	14.7	0.5	2.5	0.6	0.2	0.1	0.1	0.2	0.5	2.0
E	175.6	600.1	600.1	600.1	600.1	0.1	600.2	600.1	265.2	600.1	464.2
F	1.4	2.0	0.9	7.8	0.6	0.4	15.6	1.4	0.9	10.5	4.2
G	1.3	3.3	17.5	6.9	1.0	0.2	13.6	7.2	130.2	1.3	18.3
H	1.9	1.6	2.0	1.4	12.4	0.3	3.7	44.8	1.9	1.1	7.1
I	2.3	3.4	79.6	2.1	2.1	0.6	4.8	2.2	1.7	1.2	10.0
J	inf	0.4	0.4	37.5	inf	0.3	0.3	0.5	inf	1.0	5.8
K	44.2	1.4	17.4	0.2	0.2	0.2	0.2	0.1	0.3	0.8	6.5
L	0.1	0.1	0.2	0.1	0.2	0.1	0.3	0.1	0.1	0.1	0.1
M	600.1	600.1	68.8	600.1	600.2	0.8	600.1	1.0	600.3	mem	407.9
N	0.4	0.2	0.2	1.1	0.4	0.4	inf	0.3	inf	0.4	0.4
O	600.1	1.8	264.5	1.5	0.4	0.3	2.1	451.0	0.7	528.7	185.1
B'	0.3	0.4	0.4	0.2	0.2	0.5	0.4	0.5	0.4	0.6	0.4
C'	0.2	0.3	0.2	0.2	0.2	0.1	0.2	0.2	0.3	0.3	0.2
D'	0.2	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.2
E'	600.1	600.1	600.1	600.1	36.0	0.2	600.1	4.4	600.1	43.7	368.5

Legend: inf - infeasible, mem - out of memory status

Service strategies We use scenarios B to D to determine the impact of the chosen service strategy. We compare them to scenario K instead of scenario A, since they have 7 straddle carriers available. Scenario K applies all four service strategies. Scenarios B to D apply service strategy 4 for transport mode 4 and the same service strategy for transport modes 1, 2 and 3: strategy ded$|r_v, d_v|\sum C_v$ for scenario B, ded$|r_v, d_v|-$ for scenario C and ded$|r_v, d_v|\sum U_c$ for scenario D.

Scenario B is always solved faster than scenario C. In both cases, vehicles have to be served within their time windows, but for scenario B costs occur for each period a vehicle spends at the terminal. These costs accelerate the solution process. They probably reduce the search space by reducing the number of symmetric allocations. Scenarios B and D are solved in similar times, except for instance II which is solved faster for scenario B and scenario IX which is solved much faster for scenario D.

Comparing the results of scenarios C and K shows again that the problem is solved faster if the delay of a vehicle or unexecuted tasks are penalized in the objective function. For most

instances, results for K are similar to results for B and D. Applying several service strategies rather than a single one may be advantageous in terms of solution time (C) or have almost no impacts (B and D).

Volume of transport modes 1 to 3 We increase the volume to be handled for vehicles of transport modes 1 to 3. We double the number of vehicles per transport mode (scenario E) and the number of containers per vehicle (scenario F). By doubling the number of vehicles the problem is more difficult to solve since the number of variables is doubled and more possible allocations have to be compared. If we double the number of tasks per vehicle the solution time increases only slightly.

Volume of transport mode 4 We vary the number of vehicles of transport mode 4 arriving at the terminal from 200 (scenario A) to 100 (scenario G), to 400 (scenario H) and to 800 (scenario I). No general decrease or increase in solution time occurs for changing number of vehicles. The solution time reaches its peak for a certain number of arriving vehicles and gets smaller for fewer or more vehicles. The number of available straddle carriers is adapted to the truck workload. The number of available straddle carriers changes from 6 (scenario A) to 5 (scenario G), to 9 (scenario H) and to 10 (scenario I). These straddle carriers are also used to serve transport modes 1 to 3 and change the structure of the solution completely.

Available straddle carriers We vary the number of available straddle carriers for the same workload from 6 (scenario A) to 5 (scenario J), to 7 (scenario K) and to 8 (scenario L). For scenario J two instances become infeasible. The solution time reaches its peak for a given number of straddle carriers and is solved faster for fewer or more available straddle carriers. Probably, the problem is solved quickly if few straddle carriers are available and only few allocations are feasible or if a lot of straddle carriers are available and the optimal allocation is found very quickly. The problem gets more difficult to solve if a lot of feasible allocations exist.

Number of time periods For scenario M, we double the number of time periods describing the working day. Only 3 instances could be solved within the given time limit. By doubling the number of time periods, the number of variables and the length of vehicle time windows are also doubled. Consequently, more possible straddle carrier allocations have to be compared during the solution process and the problems gets harder to solve.

Length of vehicle time windows We shorten the length of vehicle time windows from 7 time periods (scenario A) to 5 (scenario N) and enlarge it to 9 (scenario O). If the time window is shortened to 5 time periods instances VII and IX become infeasible. The other instances are solved very quickly. If the time window is enlarged to 9 time periods the solution time of most of the instances increases. Only instances VII and IX - the two long-running instances in scenario A - do not follow this rule. Apparently, a shorter time window accelerates the solution process since fewer feasible straddle carrier allocations exist and larger time windows slow down the solution process because more alternatives exist.

Sharing straddle carriers We allow sharing straddle carriers among vehicles of the same transport mode for scenarios B', C', D' and E'. Scenarios B', C' and D' are solved very quickly. Differences are especially big for scenario C' which solves more instances than C

and solves them faster as well. Dedicating or sharing straddle carriers has less impact for scenarios B' and D'. Scenarios E' and E solve different instances; in total scenario E' solves one instance more than scenario E. Probably, the benefits of sharing straddle carriers are more important for a bigger pool of vehicles.

Other remarks The analysis above shows that the different input parameters are related. With the given results no simple rules to predict their impacts on the solution time can be established. We also notice that instance VI is solved in less than one second for all scenarios A to O. A similar behavior is not observed for another instance. This suggests that this instance has some specifications that make it easy to solve. For the moment, we were not able to determine this attribute.

3.4 Complexity analysis

Our objective is to allocate straddle carriers to different vehicles in order to reduce delays at the terminal. If all vehicles are served with the same service strategy, our allocation problem generalizes some well-known scheduling problems with known complexity.

Table 3.4: Service strategies with their equivalent scheduling problems and complexity classes

Problem formulation	Equivalent scheduling problem	Complexity
$\alpha\|r_v, d_v = T\|\sum C_v$	$1\|r_j$,pmtn$\|\sum C_j$	P
$\alpha\|r_v, d_v\|\sum C_v$		
$\alpha\|r_v, d_v = T\|\sum w_v C_v$	$1\|r_j$,pmtn$\|\sum w_j C_j$	NP-hard
$\alpha\|r_v, \tilde{d}_v, d_v = T\|\sum T_v$	$1\|r_j$,pmtn$\|\sum T_j$	NP-hard
$\alpha\|r_v, \tilde{d}_v, d_v = T\|\sum w_v T_v$	$1\|r_j$,pmtn$\|\sum T_j$	NP-hard
$\alpha\|r_v, d_v\|\sum U_c$	$1\|r_j, p_j = p\|\sum U_j$	P
$\alpha\|r_v, d_v\|-$	$1\|r_j, p_j = p\|\sum U_j$	P
$\alpha\|r_v, d_v\|\sum S_v$		
$\alpha\|r_v, d_v = T$,non-incr$\|\sum C_v$	$1\|r_j\|\sum C_j$	NP-hard
$\alpha\|r_v, d_v$,non-incr$\|\sum C_v$		
$\alpha\|r_v, d_v = T$,non-incr$\|\sum w_v C_v$	$1\|r_j\|\sum w_j C_j$	NP-hard
$\alpha\|r_v, \tilde{d}_v, d_v = T$,non-incr$\|\sum T_v$	$1\|r_j\|\sum T_j$	NP-hard
$\alpha\|r_v, \tilde{d}_v, d_v = T$,non-incr$\|\sum w_v T_v$	$1\|r_j\|\sum T_j$	NP-hard
$\alpha\|r_v, d_v$,non-incr$\|\sum U_c$	$1\|r_j\|U_j$	NP-hard
$\alpha\|r_v, d_v$,non-incr$\|$-	$1\|r_j\|U_j$	NP-hard
$\alpha\|r_v, d_v$,non-incr$\|\sum S_v$		

Consider a scheduling problem where a set of n jobs has to be processed on a single machine which is always available. Each job j has a processing time p_j, a release date r_j and a due date d_j. The machine can process one job at a time and the processing of each

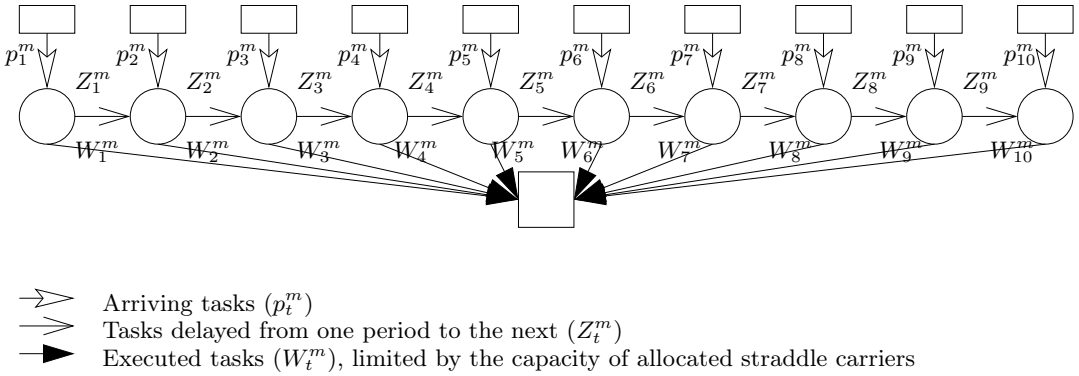


Figure 3.2: Scheme of the aggregated network flow model

job cannot start before its corresponding release time. To transform the scheduling problem to our allocation problem with a single service strategy ($m = 1$) set $s_t = 1$, $\forall t = 1, \ldots, T$, $I^m = n$, $p_i^m = p_j$ and $h^m = 1$, $r_i^m = r_j$ and $d_i^m = d_j$. We suppose that all tasks may be executed prior to T.

The straddle carrier allocation problems are at least as hard as the equivalent scheduling problems. Table 3.4 presents allocation problems with different service strategies, the equivalent scheduling problems and (if known) their complexity indicated by Brucker and Knust. An equivalent standard scheduling problem does not exist for all service strategies and only three strategies may be solved in polynomial time.

3.5 Alternative formulation

For the model presented in Section 3.1, the problem size increases with the number of vehicles arriving at the terminal as a network flow model is created for each vehicle. In this section, we present an alternative formulation where all vehicles of the same transport mode are represented via one aggregated network flow model. We discuss which service strategies qualify for such an aggregation and present their formulations as linear mixed integer programs.

3.5.1 Aggregated network flow model

The aggregated network flow model reduces the problem size as it represents all vehicles of the same transport mode by one network flow model. Figure 3.2 represents the idea of this aggregated formulation. Vehicles are transformed into tasks arriving over time and straddle carriers are allocated to tasks and not to vehicles. The optimal solution of the aggregated model indicates the number of straddle carriers to allocate to each transport mode without specifying single vehicles. To apply the aggregated model straddle carriers have to be shared among vehicles of the same transport mode and it has to be possible to disaggregate the obtained solution into an optimal allocation to single vehicles.

The aggregated formulation remains similar to the one presented in Section 3.1 with the difference that vehicle specific parameters, variables and constraints are summed up to represent all vehicles of the same transport mode.

p_t^m Number of tasks from all vehicles i of transport mode m arriving in period t
$p_t^m = \sum_{i=1|r_i^m=t}^{I^m} p_i^m \qquad \forall m = 1,\dots,M, t = 1,\dots,T$

X_t^m Number of straddle carriers allocated to transport mode m in period t
$X_t^m = \sum_{i=1}^{I^m} X_{i,t}^m \qquad \forall m = 1,\dots,M, t = 1,\dots,T$

W_t^m Number of tasks executed for transport mode m in period t
$W_t^m = \sum_{i=1}^{I^m} W_{i,t}^m \qquad \forall m = 1,\dots,M, t = 1,\dots,T$

Z_t^m Number of non-executed tasks for transport mode m in period t which are transferred to period $t+1$
$Z_t^m = \sum_{i=1}^{I^m} Z_{i,t}^m \qquad \forall m = 1,\dots,M, t = 1,\dots,T$

We reformulate the model with these parameters and variables. Constraint (3.29) describes the relation between allocated straddle carriers and handled tasks. Constraints (3.30) and (3.31) describe flow balance constraints for arriving, executed and delayed tasks. They also impose that no straddle carriers are allocated to a task prior to its arrival[3]. Constraint (3.32) imposes that all tasks are executed at the end of the time horizon. Constraint (3.33) guarantees that the number of allocated straddle carriers does not exceed the number of straddle carriers available at the terminal. Constraint (3.34) renders the solution more comprehensible by imposing that each allocated straddle carrier executes at least one task. Constraint (3.35) imposes that straddle carriers are not shared among different transport modes. Constraint (3.36) defines variable domains.

$$W_t^m \leq h^m \cdot X_t^m \qquad \forall m = 1,\dots,M, t = 1,\dots,T \tag{3.29}$$

$$Z_t^m = p_t^m - W_t^m \qquad \forall m = 1,\dots,M, t = 1 \tag{3.30}$$

$$Z_t^m = Z_{t-1}^m + p_t^m - W_t^m \qquad \forall m = 1,\dots,M, t = 2,\dots,T \tag{3.31}$$

$$Z_T^m = 0 \qquad \forall m = 1,\dots,M \tag{3.32}$$

$$\sum_{m=1}^{M} X_t^m \leq s_t \qquad \forall t = 1,\dots,T \tag{3.33}$$

$$W_t^m \geq h^m \cdot (X_t^m - 1) + 1 \qquad \forall m = 1,\dots,M, t = 1,\dots,T \tag{3.34}$$

$$X_t^m \in \mathbb{N}^+ \qquad \forall m = 1,\dots,M, t = 1,\dots,T \tag{3.35}$$

$$W_t^m, Z_t^m \in \mathbb{R}^+ \qquad \forall m = 1,\dots,M, t = 1,\dots,T \tag{3.36}$$

3.5.2 Adapted aggregated network flow models

Only service strategies shar$|r_v, d_v = T|\sum C_v$, shar$|r_v, d_v|-$ and shar$|r_v, d_v|\sum U_c$ may be represented in an aggregate way. This matches the findings in Section 3.4 where we have seen that these strategies can be solved in polynomial time. This section presents adaptations to represent those service strategies. Let $\{1,\dots,m'\}$ be the transport modes for which the specified service strategy is applied.

[3]Constraint (3.9) is thus implicitly considered here.

shar$|r_v, d_v, p_v = 1, \beta'|\sum C_v$ We want to minimize the time a vehicle spends at the terminal (waiting and service times). Since each vehicle requires exactly one task to be executed, the number of delayed tasks is identical to the number of delayed vehicles. The objective function penalizes the delay of each vehicle.

$$\min \sum_{m=1}^{m'} \sum_{t=1}^{T} Z_t^m$$

Since all vehicles are identical in terms of execution time and delay costs, every allocation where no straddle carrier remains (partly) idle while tasks have to be served is optimal. Applying the FIFO rule to the obtained aggregated allocation yields an optimal allocation to single vehicles.

shar$|r_v, d_v \beta'|\sum U_c$ We want to minimize the number of container movement requests that remain unexecuted at the departure of the vehicle. Constraints (3.30) to (3.32) have to be modified to include the possibility to leave some tasks unexecuted. Constraints (3.37) and (3.38)describe flow constraints. Constraints (3.39) and (3.40) make sure that all tasks of each vehicle are served within its time window or counted as unexecuted.

$$\min \sum_{m=1}^{m'} \sum_{t=1}^{T} U_t^m$$

$$Z_t^m = p_t^m - W_t^m - U_t^m \qquad \forall m = 1, \ldots, m', t = 1 \tag{3.37}$$

$$Z_t^m = Z_{t-1}^m + p_t^m - W_t^m - U_t^m \qquad \forall m = 1, \ldots, m', t = 2, \ldots, T \tag{3.38}$$

$$Z_T^m - U_T^m = 0 \qquad \forall m = 1, \ldots, m' \tag{3.39}$$

$$\sum_{t=t_1}^{t_2} W_t^m + U_t^m \geq \sum_{\substack{i=1,\ldots,I^m \\ |t_1 \leq r_i \wedge d_i \leq t_2}} p_i^m \qquad \begin{array}{l} \forall m = 1, \ldots, m', t_1 = 1, \ldots, T, \\ t_2 = t_1 + 1, \ldots, T \end{array} \tag{3.40}$$

All tasks have the same delay costs. To obtain an optimal allocation of straddle carriers to single vehicles, we use the earliest due date strategy to allocate available straddle carriers and unexecuted tasks to single vehicles.

shar$|r_v, d_v, \beta'|-$ We want to obtain an allocation where each vehicle is served within its time window. The problem may be solved by solving shar$|r_v, d_v, \beta'|\sum U_c$; either an allocation where no tasks remain unexecuted exists or the problem is infeasible. Allocating available straddle carriers with the earliest due date strategy to single vehicles results in an optimal allocation for single vehicles.

Additional constraints

Few additional constraints may be represented with the aggregated model.

shar$|r_v|\gamma'$ Constraints (3.30) and (3.31) already prevent allocation of straddle carriers to periods prior to the arrival of the vehicle.

shar$|\max_m|\gamma'$ The number of tasks that can be executed per transport mode per period are limited due to space or capacity restrictions. This limit is expressed by parameter q^m

$$W_t^m \leq q^m \qquad \forall m = 1, \ldots, m', t = 1, \ldots, T \tag{3.41}$$

3.5.3 Model comparison

Representing trucks in an aggregated way promises most benefits since few trains, barges and vessels, but hundreds of trucks arrive at a terminal within one working day. We suppose that trucks are served with shar$|r_v, p_v = 1, d_v = T|\sum C_v$. This section shows that aggregating vehicles for this service strategy reduces the problem size, as well as the solution time.

We implement Model 2A which is identical to Model M2 (defined on page 32) except for transport mode 4 which is modeled in an aggregated way. Table 3.5 compares the model sizes corresponding to transport mode 4. For the vehicle model, the number of variables and constraints increases with the number of vehicles (I) and the number of time periods (T). The aggregated formulation is more compact and the number of variables and constraints increases only with the number of time periods (T). With an increasing number of vehicles, the difference between both models becomes more significant.

To compare runtimes of the vehicle model and the aggregated model we execute model M2A on the instances presented in section 3.3.1 with a time limit of 600 seconds. Table 3.6 compares Model M2 and Model M2A. It indicates the number of feasible instance, the number of instances solved within 600 seconds for each model and average solution time for solved instances. It compares run times of both models via the relative gap in percentage (M2A/M2)and the absolute gap (M2 - M2A) in seconds. Gaps for solved instances for scenarios M and O are not displayed as the number of solved instances differ. It also reports the average gap between the current solution and the lower bound for unsolved instances.

Both models solve the same number of instances for almost all scenarios; only for scenarios M and O the vehicle model solves one instance more. The aggregated model solves almost all scenarios faster than the vehicle model - on average instances are solved 30% or 23 seconds faster. Especially for scenario E, the difference between both models is very big. This is quite surprising because scenario E increases the number of vehicles of transport modes 1 to 3 and does not change transport mode 4. On the other hand, only small differences occur for scenarios H and I where the number of vehicles of transport mode 4 is increased

Table 3.5: Problem size for vehicle and aggregated models for shar$|r_v, p_v = 1, d_v = T|\sum C_v$

	Vehicle model (Model M2)	**Aggregated model (Model M2A)**
Continuous variables	$3 \cdot I \cdot T$	$3 \cdot T$
Binary variables	0	0
Integer variables	0	0
Constraints	$4 \cdot I \cdot T + I + T$	$4 \cdot T + 1$

Table 3.6: Performance comparison of the vehicle model (M2) and the aggregated model (M2A)

Scenario	Feas.	Solved inst.		Avg. CPU time [s]				Avg. gap [%]	
		M2	M2A	M2	M2A	gap [%]	gap [s]	M2	M2A
A	10	10	10	13.7	0.1	0.7	13.6	-	-
B	10	10	10	30.4	0.3	1.0	30.1	-	-
C	10	8	8	10.2	1.4	13.7	8.8	21.7	18.6
D	10	10	10	2.0	3.6	180.0	-1.6	-	-
E	10	3	3	147.0	2.7	1.8	144.3	9.4	8.2
F	10	10	10	4.2	0.2	4.8	4.0	-	-
G	10	10	10	18.2	2.7	14.8	15.5	-	-
H	10	10	10	7.1	5.3	74.6	1.8	-	-
I	10	10	10	10.0	0.2	2.0	9.8	-	-
J	7	7	7	5.8	0.7	12.1	5.1	-	-
K	10	10	10	6.5	1.0	15.4	5.5	-	-
L	10	10	10	0.2	0.1	50.0	0.1	-	-
M	10	3	2	23.5	2.9	-	-	4.0	3.0
N	8	8	8	0.4	0.2	50.0	0.2	-	-
O	10	9	8	139.0	46.1	-	-	9.1	6.3
Avg.		8.5	8.4	27.9	4.5	32.4	18.2	11.0	9.0

to 400 and 800. Probably the straddle carrier allocation to vehicles of transport mode 4 on its own does not have big impacts on the solution time. But, in combination with other transport modes, the aggregating transport mode 4 helps to eliminate symmetric solutions and shortens the solution process.

3.6 Conclusion

This chapter introduced a linear mixed integer programming model for the straddle carrier allocation problem (SCAP) at intermodal container terminals. It allocates straddle carriers to different transport modes with the objective to minimize overall delays at the terminal. We presented a core linear mixed integer programming model where the service of a vehicle is modeled as a network flow. Arriving, executed and delayed tasks are flows in the network and the capacity of allocated straddle carriers limits the number of executed tasks via the associated arc capacities. We presented parameters, variables and constraints that are necessary to adapt the core model to different service strategies. The modular structure of these vehicle models model makes it possible to represent terminals using different service strategies for different transport modes.

We carried out a sensitivity analysis to analyze the impact of input parameters on the solution time. Input parameters are highly interrelated and it was not possible to determine general rules to describe the influence of all input parameters on the solution time. We observed that a high number of time periods increases the solution time considerably and

that the impact of the number of vehicles depends on the chosen service strategy.

We showed how to transform the straddle carrier allocation problem serving all vehicles with the same strategy into classical scheduling problems. We used this analogy to state the complexity of the straddle carrier allocation problem for different service strategies. It appears that most of the analyzed problems are NP-hard.

For some service strategies, all vehicles of the same transport mode may be represented by one aggregated model. We implemented all possible service strategies as mixed integer programs and discussed how the aggregated solution may be transformed into an allocation to single vehicles. Aggregating the model reduces the problem size as the number of variables depends no longer on the number of vehicles. Experiments show that the aggregated model outperforms the vehicle model with regard to computation time.

Several possibilities exist to continue our work. Firstly, develop the presented model further by including more details or representing other service strategies. Secondly, include stochasticity in the model or analyze the performance of the proposed allocation in a stochastic environment. Thirdly, combine the straddle carrier allocation problem with interrelated optimization problems.

Our linear mixed integer programming model represents the most important characteristics of the straddle carrier allocation problem. More details can be added by differentiating import, export, empty, reefer and hazardous containers with their specific handling requirements, service strategies and delay costs. The model can also be extended to define the allocation in accordance with union agreements. For the current model implementation, each task has to be finished in the time period in which it has been started. If the handling capacity per straddle carrier per integer is real (and not integer like in our case) the model has to be adapted to enable tasks to be served over two periods.

Our optimization model uses average handling times whereas real handling times depend on the arrival pattern of trucks, the storage locations of containers, the scheduling of tasks and the straddle carrier routing. It would be interesting to see how the allocation proposed by the optimization model performs in a stochastic situation - like we do in Chapter 4 - or to include dynamic aspects directly into the optimization model. The deterministic optimization model could be combined with simulation or queuing models to include a nonlinear relation between the number of allocated straddle carriers and the number of executed tasks.

A broader study (e.g., via simulation) could evaluate the impacts of different straddle carrier and storage allocation strategies on straddle carrier scheduling and routing and on handling times. This would provide more insights on the interconnection between these problems and quantify how they influence each other. These findings could then be used to improve the allocation proposed by the optimization problem.

Chapter 4

Case study: Grand Port Maritime de Marseille

Container terminals are highly complex and uncertain environments: exact arrival times and volumes of vessels, trucks, trains and barges are not known in advance or differ from the announced values and travel and handling times of straddle carriers vary among jobs. When planning the straddle carrier allocation for the next day, exact arrival and handling times are unknown. Simulation can include this unreliability easily, but is impracticable when dealing with a large number of alternatives. We use the model based on average handling times presented in Chapter 3 to determine an allocation and use simulation to evaluate this allocation in a stochastic environment.

This chapter presents a case study carried out for a container terminal at the Grand Port Maritime de Marseille. Section 4.1 presents the harbor of Marseilles and the analyzed container terminal. Section 4.2 formulates the mixed integer programming model for this container terminal, shows that the resulting allocation problem is NP-hard and presents allocation obtained for real world instances. Section 4.3 presents and validates the simulation model representing the analyzed container terminal. Experiments evaluate the quality of the allocation proposed by the optimization model for different levels of variability. Section 4.4 concludes the chapter.

4.1 Situation at Marseilles

The Grand Port Maritime de Marseille (GPMM) in France is a general cargo seaport. It deals with various types of traffic including crude oil and oil products (oil, gas and chemical products), general cargo (containers and other packaging), dry bulk (minerals and cereals) and liquid bulk (chemicals and food), as well as passenger traffic. The port features two harbors: the "East Harbor" within the city of Marseilles and the "West Harbor" at Fos at 70km from Marseilles. GPMM is the largest port in France and among the largest in Europe in terms of total throughput. It is especially active in oil cargo and is the third oil port worldwide (European Sea Ports Organisation (2010)).

Three container terminals are operating at GPMM: Eurofos and Seayard at the West Harbor and Intramar at the East Harbor. Most of the container activity takes place at the West Harbor. Altogether the three container terminals handled 1 060 000 TEUs in 2012,

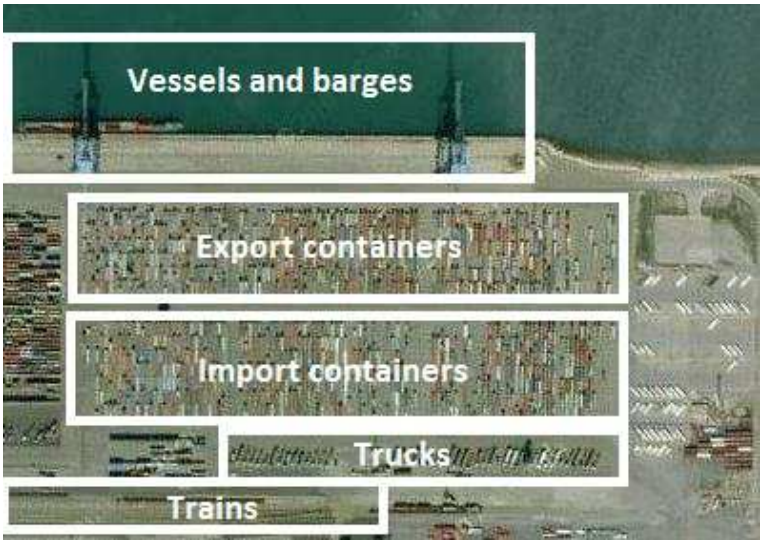


Figure 4.1: Bird's view of container terminals at GPMM

12% more than in 2009. The majority of inland transportation is done via trucks (80%) and only a small part by trains (8.3%) or barges (7.4%).

GPMM aims to increase its volume of containerized cargo. Several projects were started to increase the handling capacity of the container terminals. Through expansion the handling capacity was doubled to 2 000 000 containers per year in 2012. In addition, Hutchison Port Holdings (HPH) will start operating at GPMM in 2018 with an expected increase of handling capacity of 1 500 000 TEU per year. To keep pace with the increased volumes, GPMM also invests in infrastructure: on the seaside new super post panamax quay cranes were installed; on the landside a new highway and a faster connection between the Rhone canal and the container terminals will be constructed and the terminals will be connected to freight high speed lines.

4.1.1 Applied service strategies

Our study is based on one of the terminal at GPMM. This terminal serves maritime vessels, trucks, trains and barges and operates only with straddle carriers. Figure 4.1 shows a bird's view of this terminal (and parts of the adjacent one). The different areas where vessels, barges, trains and trucks are unloaded and loaded, as well as the storage areas for import and export containers, are perceptible. Import containers are stored close to the quay and export containers close to the truck and train exchange areas to minimize travel times. The dockers at GPMM work according to shifts with fixed starting and ending times and have to be booked the day before. At present, straddle carriers are either allocated to vessels or shared among inland transport modes (truck, train, barge). The sequence of containers to be handled is determined by their terminal operating system.

Vessels (mode $m = 1$ in the sequel) are unloaded and loaded at the quay by quay cranes. The quay crane throughput imposes the number of containers to be transported. Vessels have to be completely served within their time windows which are defined by the terminal operator. Straddle carriers are allocated to exactly one quay crane and are evenly distributed among the vessel. No additional straddle carriers are allocated to a vessel once its service has

started. But, superfluous straddle carriers may be retrieved. With the notation introduced in Section 2.1, the service of vessels is described by ded$|r_v, d_v$,non-incr, $\max_v|$-.

Barges (mode $m = 2$) are served at the same quays as vessels by one quay crane. The number of containers to be transported depends on quay crane throughput. Barges should be unloaded and loaded as fast as possible. But in reality, they are only served if the equipment is not used by vessels. Hence, the ready and departure times of barges do not correspond to their arrival and scheduled departure times, but on the availability of the quay cranes. Straddle carriers are allocated to exactly one barge and are not shared among barges. The service of barges is described by ded$|r_v, d_v, \max_v|\sum C_v$.

Trains (mode $m = 3$) are served at the rail station. Railcars stay during a fixed time window at the terminal and are picked up by an engine according to a fixed schedule every day. Penalties have to be paid for every container that is not loaded or unloaded when the train leaves the terminal. Straddle carriers transport containers between the yard and the buffer in front of the railcars. Loading and unloading to and from railcars is done by reach stackers. Reach stackers are not represented as they are operated by different dockers and do not represent a bottleneck. Straddle carriers are shared among all trains. The service of trains is described by shar$|r_v, d_v|\sum U_c$.

Trucks (mode $m = 4$) should be served as fast as possible, but at least at the end of the working day. At the entering gate, trucks are assigned to parking slots where they are loaded and unloaded directly by straddle carriers. Straddle carriers are shared among all trucks. The service of trucks is described by shar$|r_v, d_v = T|\sum C_v$.

4.1.2 Instances

We generated 10 instances from the historic data observed at the container terminal. An instance represents the 14 hours of a working day where all four transport modes are served at the terminal. The time horizon is divided into one-hour intervals to represent the fixed allocation at the terminal where straddle carriers can be reallocated at most every hour. The terminal operator also provided the average number of tasks straddle carriers ($h^1 = h^2 = h^3 = 7$ and $h^4 = 10$) and quay cranes ($q^1, q^2 = 20 \cdot$ nb. cranes) can handle per hour. More tasks can be handled for trucks as storage requests can be combined with retrieval requests; this is not possible for trains, barges and vessels as the loading operation starts only when the unloading operation is finished.

The terminal operator provided historic data about his workload and we chose ten days among them. For trains, barges and vessels the arrival and due dates as well as the number of containers to be loaded and unloaded equal the historic data. For trucks, the average numbers of import and export containers per period are extracted from historic data. Table 4.1 indicates the volumes to be handled per day and per transport mode: 1 or 2 vessels, 0 or 1 barges, 0 or 1 trains and 279 to 769 truck containers have to be served per day.

4.2 Optimization model

This section combines the elements presented in Chapter 3 to model the situation at the container terminal at Marseilles. The problem is NP-hard since the service strategies of vessels is NP-hard (see Section 3.4).

Table 4.1: Instances representing the workload (in containers to be handled) at the terminal for trucks, trains, barges and vessels

Day	1	2	3	4	5	6	7	8	9	10
Vessel 1	285	477	126	137	191	375	169	266	351	627
Vessel 2	-	-	282	-	-	-	192	575	-	195
Barge	-	66	-	115	64	-	37	-	45	-
Train	16	15	-	26	45	23	20	54	23	37
Trucks	279	486	371	405	380	482	391	521	769	477
Total	580	1044	779	683	680	880	809	1416	1188	1336

Key: - No such vehicle arrives at the terminal

4.2.1 Problem formulation as MIP

Independent submodels are implemented for each transport mode to represent the chosen service strategies. Table 4.2 presents the parameters, indices and variables used to model the container terminal. For vessels ($m = 1$), barges ($m = 2$) and trains ($m = 3$), one network flow is implemented for each vehicle. Trucks ($m = 4$) are aggregated and tasks are represented by a single network flow model to reduce the solution size and speed up the solution process.

The objective is to serve each vessel within its time window while minimizing the number of periods a barge has to spend in the terminal, the number of non-executed tasks at the departure of a train and the number of trucks delayed from one period to the next.

$$w_2 \cdot \sum_{i=1}^{I^2} \sum_{t=r_i^2}^{T} Y_{i,t}^2 + w_3 \cdot \sum_{i=1}^{I^3} U_i^3 + w_4 \cdot \sum_{t=1}^{T} Z_t^4$$

s.t.

Constraints limiting the number of executed tasks by the capacity of allocated straddle carriers

$$W_{i,t}^m \leq h^m \cdot X_{i,t}^m \qquad m = 1,2,3, \forall i = 1,\ldots,I^m, t = 1,\ldots,T \tag{4.1}$$

$$W_t^m \leq h^m \cdot X_t^m \qquad m = 4, \forall t = 1,\ldots,T \tag{4.2}$$

Constraints imposing that each allocated straddle carrier executes at least one task

$$W_{i,t}^m \geq h^m \cdot (X_{i,t}^m - 1) + 1 \qquad m = 1,2,3, \forall i = 1,\ldots,I^m, t = 1,\ldots,T \tag{4.3}$$

$$W_t^m \geq h^m \cdot (X_t^m - 1) + 1 \qquad m = 4, \forall t = 1,\ldots,T \tag{4.4}$$

Constraints limiting the number of executed tasks per period to the quay crane capacity

$$W_{i,t}^m \leq q_i^m \qquad m = 1,2, \forall i = 1,\ldots,I^m, t = 1,\ldots,T \tag{4.5}$$

Table 4.2: Parameters, indices and variables used to model the straddle carrier allocation problem for the given container terminal at Marseille

Parameters and indices:

T	Number of time periods describing the time horizon
M	Number of transport modes being served at the terminal with $m = 1$ vessels, $m = 2$ barges, $m = 3$ trains and $m = 4$ trucks
I^m	Number of vehicles of transport mode m arriving during the time horizon
t	Index of a time period, $t = 1, \ldots, T$
m	Index of a transport mode, $m = 1, \ldots, M$
i	Index of a vehicle of transport mode m, $i = 1, \ldots, I^m$
r_i^m	Period t in which vehicle i of transport mode m arrives at the terminal
d_i^m	Period t in which vehicle i of transport mode m leaves the terminal ($d_i^m \leq T$)
p_i^m	Total number of tasks to be carried out for vehicle i of transport mode m
p_t^m	Total number of tasks arriving for transport mode m in period t
q_i^m	Maximum number of tasks that can be executed per vehicle i of transport mode m per period
s_t	Number of available straddle carriers per period t
h^m	Average number of tasks a straddle carrier serving transport mode m can handle per period, we assume that $h^m \in \mathbb{N}^+$, $\forall m \in \mathcal{M}$ and that $h^m \geq 1$
w_i^m	Penalty for delaying vehicle i of transport mode m

Variables:

$X_{i,t}^m$	Number of straddle carriers allocated in period t to vehicle i of transport mode m
X_t^m	Number of straddle carriers allocated in period t to transport mode m
$W_{i,t}^m$	Number of tasks executed in period t for vehicle i of transport mode m depending on the number of allocated straddle carriers
W_t^m	Number of tasks executed in period t for transport mode m depending on the number of allocated straddle carriers
$Z_{i,t}^m$	Number of non-executed tasks in period t for vehicle i of transport mode m which are transferred to period $t + 1$
Z_t^m	Number of non-executed tasks in period t for transport mode m which are transferred to period $t + 1$
U_i^m	Number of tasks which are not executed for vehicle i of transport mode m at its departure
$Y_{i,t}^m$	Binary variable indicating if vehicle i of transport mode m is completely served at the end of period t
$D_{i,t}^m$	Binary variable indicating if the service of vehicle i of transport mode m has already been started in $t - 1$

Mass balance constraints for arrived, executed and delayed tasks

$$Z_{i,t}^m = p_i^m - W_{i,t}^m \qquad m = 1,2,3, \forall i = 1,\dots,I^m, t = r_i^m \tag{4.6}$$

$$Z_{i,t}^m = Z_{i,t-1}^m - W_{i,t}^m \qquad m = 1,2,3, \forall i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \tag{4.7}$$

$$Z_t^m = p^m - W_{,}^m \qquad m = 4, t = 1 \tag{4.8}$$

$$Z_t^m = Z_{t-1}^m - W_{i,t}^m \qquad m = 4, \forall t = 2,\dots,T \tag{4.9}$$

Constraints imposing trucks, barges and vessels are served prior to their deadlines

$$Z_{i,d_i^m}^m = 0 \qquad m = 1,2, \forall i = 1,\dots,I^m \tag{4.10}$$

$$Z_T^m = 0 \qquad m = 4 \tag{4.11}$$

Constraints determining delays of barges and trains

$$Y_{i,t}^m \geq \frac{Z_{i,t}^m}{p_i^m} \qquad m = 2, \forall i = 1,\dots,I^m, t = r_i^m,\dots,T \tag{4.12}$$

$$Z_{i,d_i^m}^m - U_i^m = 0 \qquad m = 3, \forall i = 1,\dots,I^m \tag{4.13}$$

Constraints imposing a non-increasing service for vessels

$$D_{i,t}^m \geq \frac{p_i^m - Z_{i,t-1}^m}{p_i^m} \qquad m = 1, \forall i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \tag{4.14}$$

$$X_{i,t}^m \leq X_{i,t-1}^m - s_t \cdot (D_{i,t}^m - 1) \qquad m = 1, \forall i = 1,\dots,I^m, t = r_i^m + 1,\dots,T \tag{4.15}$$

Constraints limiting the total number of allocated straddle carriers

$$\sum_{m=1,2} \sum_{i=1}^{I^m} X_{i,t}^m + X_t^3 + X_t^4 \leq s_t \qquad \forall t = 1,\dots,T \tag{4.16}$$

Constraints allowing sharing straddle carriers for trains and trucks and preventing sharing for barges and vessels

$$X_{i,t}^m \in \mathbb{N}^+ \qquad m = 1,2, \forall i = 1,\dots,I^m, t = 1,\dots,T \tag{4.17}$$

$$X_{i,t}^m \in \mathbb{R}^+ \qquad m = 3, \forall i = 1,\dots,I^m, t = 1,\dots,T \tag{4.18}$$

$$\sum_{i=1}^{I^m} X_{i,t}^m \leq X_t^m \qquad m = 3, \forall t = 1,\dots,T \tag{4.19}$$

$$X_t^m \in \mathbb{N}^+ \qquad m = 3,4, \forall t = 1,\dots,T \tag{4.20}$$

Variable domains

$$W_{i,t}^m, Z_{i,t}^m \in \mathbb{R}^+ \qquad m = 1,2,3, i = 1,\ldots,I^m, t = 1,\ldots,T \tag{4.21}$$

$$W_t^m, Z_t^m \in \mathbb{R}^+ \qquad m = 4, t = 1,\ldots,T \tag{4.22}$$

$$U_i^m \in \mathbb{R}^+ \qquad m = 3, i = 1,\ldots,I^m \tag{4.23}$$

$$D_{i,t}^m \in \{0,1\} \qquad m = 1, \forall i = 1,\ldots,I^m, t = r_i^m + 1,\ldots,T \tag{4.24}$$

$$Y_{i,t}^m \in \{0,1\} \qquad m = 2, \forall i = 1,\ldots,I^m, t = r_i^m + 1,\ldots,T \tag{4.25}$$

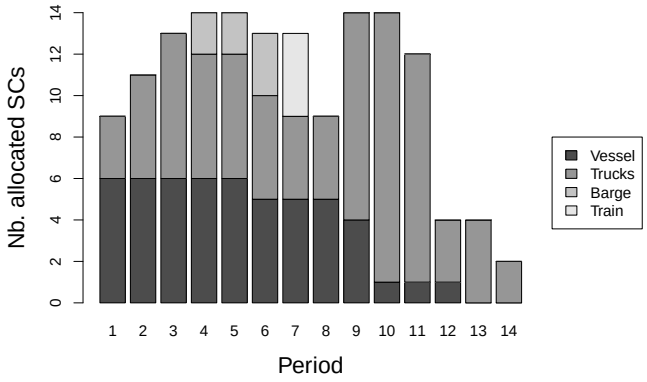
4.2.2 Numerical experiments

The optimization is implemented in and solved with IBM ILOG CPLEX 12.1. We solve each instance with a maximum of 10, 12 and 14 available straddle carriers. We set the delay costs for barges ($w^2 = 50$), trains ($w^3 = 10$) and trucks ($w^4 = 1$) so as to represent the service priorities at the terminal favoring barges and trains. With these costs and $h^4 = 7$, train tasks are not executed only if the problem becomes infeasible otherwise. We first illustrate the output of the allocation model for one instance and provide results for all instances thereafter.

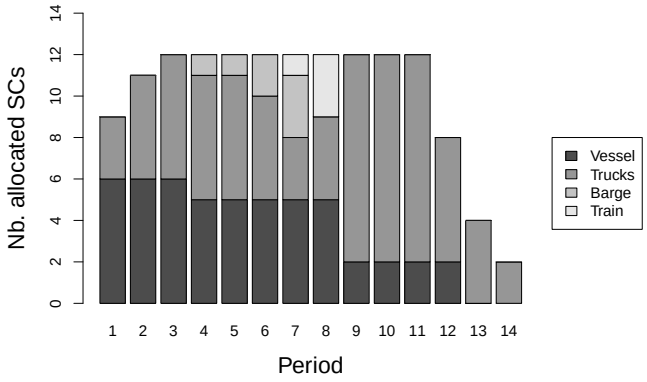
The solution of the optimization model provides information about the number of straddle carriers to allocate to each vehicle or transport mode ($X_{i,t}^m$ and X_t^m) as well as about the resulting delays for trucks ($\sum Z_t^m$), trains ($\sum U_i^m$) and barges ($\sum Y_{i,t}^m$). Figure 4.2 shows the results for 10, 12 and 14 available straddle carriers for Day 9 with one train, one barge and one vessel. It illustrates how many straddle carriers to allocate to the vessel, the trucks, the barge and the train at each of the periods 1 to 14. It also states the delays from trucks, trains and barges resulting from this allocation. The vessel is always served within its time window $(1 - 12)$.

For 14 straddle carriers, almost no delays occur: only one truck is delayed by one period. The barge spends 3 periods at the terminal which is its minimum service due to the limited quay crane throughput. All train containers are unloaded and loaded. For 12 straddle carriers, more delays occur. Especially, the service quality of trucks decreases. Fewer straddle carriers are allocated to trucks in periods 7, 10 and 11 and more straddle carriers in period 12 to serve the delayed trucks of previous periods. The barge has to spend four periods in the terminal. All train tasks are executed within two periods. The service of the vessel is partly shifted to periods 10 to 12 to free straddle carriers for the other transport modes during periods 4, 5 and 9. For 10 straddle carriers, delays occur for all transport modes. A high percentage of trucks are served after periods 13 and 14 after the vessel left the terminal. The barge spends five periods in the terminal. The train is served later and not completely. The service of the vessel is shifted as much as possible to the end of its time window to minimize impacts on other transport modes.

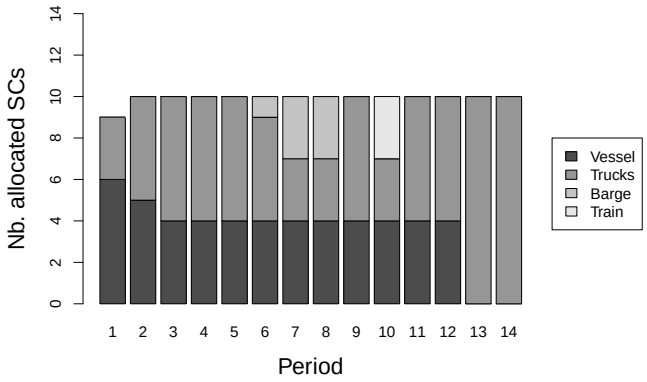
To decide how many straddle carriers to order for the next day the terminal operator executes the model for the expected workload for different numbers of available straddle carriers and chooses the number of straddle carriers with an adequate service level. In this case, he will probably chose 12 or 14 straddle carriers depending on the desired truck service



(a) 14 straddle carriers: 1 delayed truck task, 3 periods per barge, 0 unexecuted train containers



(b) 12 straddle carriers: 63 delayed truck tasks, 4 periods per barge, 0 unexecuted train containers



(c) 10 straddle carriers: 607 delayed truck tasks, 5 periods per barge, 2 unexecuted train containers

Figure 4.2: Optimal allocation and resulting delays for different numbers of available straddle carriers

Table 4.3: Solution times and delays for trucks (nb. delayed tasks), trains (nb. of non-executed tasks) and barges (nb. periods spent at the terminal) for 10, 12 and 14 available straddle carriers

s_t	Day	1	2	3	4	5	6	7	8	9	10
10	Truck	0	275	247	26	0	115	129	inf	607	inf
	Train	0	1	-	0	0	0	0	inf	2	inf
	Barge	-	5	-	5	3	-	1	inf	4	inf
	Time [s]	0.2	0.5	0.1	0.2	0.1	0.1	0.2	inf	0.3	inf
12	Truck	0	55	1	0	0	0	0	inf	63	inf
	Train	0	0	-	0	0	0	0	inf	0	inf
	Barge	-	3	-	5	3	-	1	inf	3	inf
	Time [s]	0.2	0.1	0.1	0.1	0.1	0.1	0.1	inf	0.2	inf
14	Truck	0	4	0	0	0	0	0	4	1	0
	Train	0	0	-	0	0	0	0	0	0	0
	Barge	-	3	-	5	3	-	1	-	2	-
	Time [s]	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1

Key: - No such vehicle arrives at the terminal

quality. If big changes occur with regard to the initial planning (e.g. delayed arrival of a vessel), the model can easily be adapted to the new situation and provide an updated allocation.

Table 4.3 presents the delays for all 10 days with 10, 12 and 14 available straddle carriers and indicates the time needed to solve each instance. Unsurprisingly, the number of delays increases if fewer straddle carriers are available. Some instances become even infeasible (marked by inf) because the available straddle carriers cannot execute all tasks. All instances are solved in less than a second; infeasibilty is also discovered immediately. Trucks have the lowest priority at the terminal and are the first ones to be delayed if capacity is insufficient. This explains why so many trucks are delayed while almost no delays for trains and barges occur.

4.3 Simulation model

Container terminals are highly complex and dynamic environments. Arrival times and volumes of external vehicles as well as internal handling times are subject to variability. Simulation can easily represent these interactions and uncertainty and is widely used to represent container terminals (see Angeloudis and Bell (2011) for a literature overview). Several studies (e.g., Gambardella et al.; 2001; Vis et al.; 2005; Briskorn et al.; 2006) use simulation to validate the results of their analytical models in a stochastic environment. We use simulation to evaluate the performance of the straddle carrier allocation proposed by our optimization model.

4.3.1 Discrete event simulation model

We use a terminating discrete event simulation to represent storage and transport operations within the container terminal at GPMM. The simulation model is an adapted version of the model presented in Rodriguez Verjan and Dauzère-Pérès (2010) and implemented in Arena. Its objective is to evaluate the straddle carrier allocation proposed by the optimization problem. The input parameters of the model are a straddle carrier allocation and the arriving trucks, trains, barges and vessels together with their arrival and departure times and the volumes to be handled.

All straddle carriers are assumed to be identical and can transport one container at a time. Straddle carriers are allocated to one transport mode or one vehicle for a one-hour interval. Export and import containers to be transported send transportation requests. If an allocated straddle carrier is idle, it travels to the current container position, picks it up, travels to the container destination and puts it down. Otherwise, the container waits for a straddle carrier to become idle. Since our main objective is to evaluate the proposed allocation plan, storage policies and straddle carrier routing are not modeled in detail. Travel times and times for picking up or putting down a container are uniformly distributed around average service times.

The arrival of trucks varies over the working day due to peak and low periods. We model these time dependent truck arrivals via a non-stationary Poisson process. A truck can only enter the terminal if a parking space in the loading/unloading area is free; otherwise it has to wait at the gate. Once it arrives at the parking space it requests a straddle carrier to unload the export container. A given percentage of trucks also request an import container to be loaded. Different straddle carriers may transport the export and import container for the same truck. The truck leaves the terminal as soon as it is unloaded (only export container) or loaded (import and export container). We record the number of trucks that enter and leave the terminal and the service time of each truck.

We model the arrival of a train as a uniform distribution and its volume as triangular distribution around historic data. Unloading operations begin after the arrival of the train. Straddle carriers allocated to trains transport the containers to the export area of the yard. When the unloading process is completed, the loading operation begins and allocated straddle carriers pick up import containers at the yard and transport them to the station. The train leaves the terminal at the scheduled time even if loading/unloading operations are not completed. We record the numbers of not unloaded and not loaded containers.

Vessels and barges are modeled in the same way. Their arrivals and volumes are modeled as uniform and triangular distributions around historic data. Berth and quay crane assignments are out of the scope of this study and we assume that the berth and the quay cranes are free and operational at the vehicles planned arrival. Quay cranes unload containers from the ship on the quay. If the unloading is finished they load containers from the quay into the ship. The time needed to load or unload one container is modeled as a uniform distribution about an average value. To avoid congestion at the bay, the number of containers that may be stacked in a buffer below a quay crane are limited. Quay crane activity stops if this buffer is full during unloading or empty during loading operations. Straddle carriers transport containers between the buffer and the yard. We record the number of loaded and unloaded containers for each vessel and each barge. We also record the service times of barges.

4.3.2 Model validation

Due to difficulties to obtain historic data about the allocation of straddle carriers it was not possible to validate the simulation model against real data. We chose to validate a non-stochastic version of our simulation model against the results of the optimization model. Arrival times and volumes of vehicles are set to the historic values. Straddle carrier travel and handling times are chosen to mimic the average handling capacities used in the optimization model. In this case, delays of the optimization and the simulation model should resemble.

Table 4.4 compares the results of the optimization and the simulation model for the instances presented in Section 4.1.2 with 10, 12 and 14 available straddle carriers. Delays for both models are similar for all transport modes. The same number of train tasks remain unexecuted; except for one instance. The barge service time obtained by the simulation model is more precise and almost identical to the service time indicated by the optimization model. All vessels are completely served within their time windows for both models; except for one instance.

The comparison of trucks delays is not so straightforward as the optimization and the simulation model measure truck delays in different ways. The optimization model measures the number of containers delayed from one period to the next and the simulation model the average service time per truck. To obtain the number of trucks delayed from one period to the next for the optimization model, we divide the number of delayed containers by the average load of 1.9 containers per truck. More truck delays in the optimization model find their equivalent in higher average service times and a higher standard deviation. A correlation coefficient of 0.94 with a p-value of 3.7E-9 proves that truck delays and truck service times are significantly correlated. Figure 4.3 illustrates this correlation.

Results show that the average service times per container mimic the average straddle carrier handling capacities of the optimization model. A straddle carrier needs averagely 200 seconds to travel between the yard and the loading/unloading area and averagely 60 seconds to pick up or put down a container. For trains, barges and vessels, straddle carriers are allocated to exactly one vessel and the loading process starts when the unloading process is finished. The time needed to transport one container equals the time for picking up and putting down the container and one full and one empty travel to and from the yard. The service time per container in the simulation model is $2 \cdot 200 + 2 \cdot 60 = 520$ seconds or 8.67 minutes. For $h^1 = h^2 = h^3 = 7$ the service time per container in the optimization model is $60/7 = 8.57$ minutes. For trucks, straddle carriers serve several trucks in parallel and may combine the storage of an export container with the retrieval of an import container. This suppresses one empty travel between the parking slots and the yard. Whether import and export tasks may be combined depends on the arrival of trucks and the number of allocated straddle carriers. Trucks served without delays stay averagely 12.0 minutes at the terminal. The average truck load is 1.9 containers, which leads to an average service time of $12.0/1.9 = 6.3$ minutes per container which is close to the $60/10 = 6$ minutes per container ($h^4 = 4$) in the optimization model.

We conclude that the simulation model represents the same situation as the optimization model and may be used to analyze the solution quality of the straddle carrier allocation proposed by the optimization model in a stochastic environment.

Table 4.4: Comparison of delays obtained by optimization and simulation for 10, 12 and 14 available straddle carriers (SC)

Inst.	**Optimization model**				**Simulation model**					
	Truck [delayed cont]	Train [not served cont]	Barge [service time h]	Vessels [not served cont]	Truck [service time min]	Truck [stdv. time]	Train [not served cont]	Barge [service time h]	Vessel 1 [not served cont]	Vessel 2 [not served cont]
1_10SC	0.0	0	-	0	10.8	3.6	0	-	0	-
2_10SC	144.7	1	6	0	32.2	22.5	0	5.9	0	-
3_10SC	130.0	-	-	0	44.4	43.6	-	-	0	0
4_10SC	13.7	0	6	0	11.7	5.6	0	5.7	0	-
5_10SC	0.0	0	4	0	13.1	3.9	0	3.9	0	-
6_10SC	60.5	0	-	0	18.7	10.3	0	-	0	-
7_10SC	67.9	0	2	0	24.9	17.6	0	1.9	0	0
8_10SC	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
9_10SC	319.5	2	5	0	49.7	46.2	2	4.8	0	-
10_10SC	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
1_12SC	0.0	0	-	0	idem	idem	idem	idem	idem	idem
2_12SC	28.9	0	4	0	13.1	5.8	0	3.9	0	-
3_12SC	0.5	-	-	0	11.5	3.3	-	-	0	0
4_12SC	0.0	0	6	0	11.1	3.3	0	5.7	0	-
5_12SC	0.0	0	4	0	13.1	3.9	0	3.9	0	-
6_12SC	0.0	0	-	0	12.8	3.0	0	-	0	-
7_12SC	0.0	0	2	0	13.2	4.7	0	1.9	0	0
8_12SC	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
9_12SC	33.2	0	4	0	14.2	5.5	0	3.9	0	-
10_12SC	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

Inst.	Optimization model				Simulation model					
	Truck [delayed cont]	Train [not served cont]	Barge [service time h]	Vessels [not served cont]	Truck [service time min]	Truck [Stdv. time]	Train [not served cont]	Barge [service time h]	Vessel 1 [not served cont]	Vessel 2 [not served cont]
1_14SC	0.0	0	-	0	idem	idem	idem	idem	idem	idem
2_14SC	2.1	0	4	0	11.5	3.1	0	3.9	0	-
3_14SC	0.0	-	-	0	11.0	3.2	-	-	0	0
4_14SC	0.0	0	-	0	idem	idem	idem	idem	idem	idem
5_14SC	0.0	0	-	0	idem	idem	idem	idem	idem	idem
6_14SC	0.0	0	-	0	12.8	3.0	0	-	0	-
7_14SC	0.0	0	2	0	13.2	4.7	0	1.9	0	0
8_14SC	2.1	0	-	0	14.9	7.2	0	-	1	0
9_14SC	0.5	0	3	0	11.7	3.8	0	2.9	0	-
10_14SC	0.0	0	-	0	12.2	7.2	0	-	0	0

Key: - no such vehicle arrived at the terminal idem same allocation than for instance with fewer available straddle carriers

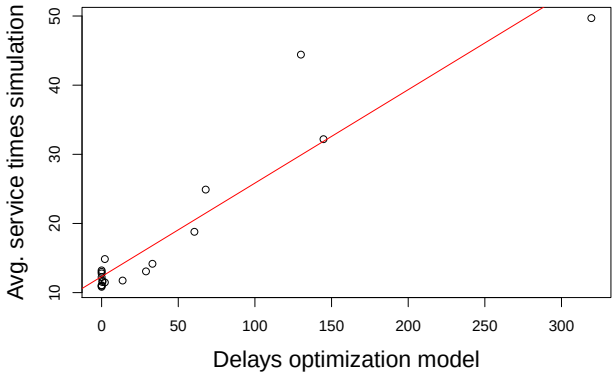


Figure 4.3: Correlation between truck delays of the optimization model and average truck service times of the simulation model

4.3.3 Numerical experiments

We use discrete event simulation to evaluate how the straddle carrier allocation proposed by the optimization model performs in a stochastic environment with uncertain arrivals, volumes and handling times. We run the simulation model with the proposed straddle carrier allocation for all instances for 10, 12 and 14 available straddle carriers with 10, 30 and 50 percent variability for handling and travel times of straddle carriers. 1000 replications are executed for each experiment. Table 4.5 summarizes the simulation input.

Table 4.5: Input parameters for simulation runs

Trucks	
Arrivals	Non-stationary Poisson process with hourly averages equal to historic arrivals
Trains, barges and vessels	
Arrivals	Uniformly distributed around the historic arrival plus/minus 15 minutes
Volumes	Triangularly distributed around the historic volume plus/minus 10 percent
Quay crane	Service time per container uniformly distributed between 2 and 3 minutes
Crane buffer	Space for 6 containers
Straddle carriers - three scenarios	
Allocation	As proposed by the optimization model
Travel times	Uniformly distributed around 3.3 minutes plus minus 10, 30 or 50 percent
Storage times	Uniformly distributed around 1.0 minute plus minus 10, 30 or 50 percent

Table 4.6: Delays for trucks, trains, barges and vessels obtained via simulation with different levels of variability for container handling

	No variability	10% variability	30% variability	50% variability
Truck [service time min]	17.37	22.33	23.65	27.50
Std Dev.	15.57	18.34	19.23	21.55
Train [not served cont]	0.11	0.11	0.12	0.23
Std Dev.	n.a.	0.23	0.33	0.60
Barge [not served cont]	0.21	0.01	0.02	0.31
Std Dev.	n.a.	0.07	0.21	0.95
Vessels [not served cont]	0.03	2.03	4.42	13.46
Std Dev.	n.a.	4.58	6.64	9.50

Table 4.6 shows average delays and standard deviations of delays for trucks, trains, barges and vessels for the deterministic simulation and simulation with variabilities of 10, 30 and 50 percent. It displays average truck service times and the number of non-executed container movements for trains, barges and vessels. For the deterministic case no standard deviation is given as the scenario is run only once. Unsurprisingly, a bigger variability of travel and handling times of straddle carriers decreases the performance at the terminal and increases the standard deviation.

The average service time of trucks increases by 27.9% for a variability of 10 percent, by 35.5% for a variability of 30 percent and by 57.2% for a variability of 50 percent. Higher variability deteriorates the service quality of vessels: the number of unexecuted tasks increases from 0 without variability to over 13 for a variability of 50 percent. For trains, impacts are less drastic: the number of not executed tasks remains almost unchanged for 10 and 30 percent variability, but delays are doubled for 50 percent variability. For barges, the stochastic cases with variability of 10 and 30 percent perform better than the deterministic case and the case with 50 percent variability slightly worse. In the deterministic case, one barge container cannot be loaded at Day 7. In the stochastic case, this container can be served most of the times. This gain offsets the small increases in delays of barges at other days.

Student's t-test show when the differences between the deterministic simulation and the simulations with 10, 30 and 50 percent variability are statistically significant. The obtained results confirm the observations above. For trucks, differences are only significant for a variability of 50 percent. For vessels, differences are significant for all three scenarios. For trains and barges, differences are not significant for all levels of variability.

To understand why vessels are more sensitive to variability than the other transport modes we compare the number of tasks to be executed with the capacity of allocated straddle carriers. Table 4.7 shows the average number of containers to be executed, the number of additional movements allocated straddle carriers could execute and the percentage of unused capacity. Trucks and barges have a excess capacity of around 10%, trains of around 13% and vessels of around 6%. A smaller spare capacity increases the impact of variability, as spare capacity counterbalances stochastic impacts. In addition, vessels (and barges) interact

Table 4.7: Volumes and unused capacity per transport mode

	Avg. demand [cont.]	Reserve capacity [cont.]	Unused capacity [%]
Trucks	470.2	46.3	9.8
Trains	25.9	3.4	13.1
Barges	57.3	5.6	9.9
Vessels	284.0	18.0	6.3

with quay cranes which amplifies the negative impacts of variability. Straddle carriers are shared among trucks and delays are passed on later trucks; this also increases the impact of variability.

4.4 Conclusion

This chapter presented a case study for the straddle carrier allocation problem carried out for a container terminal at the Grand Port Maritime de Marseille. We used the modeling elements introduced in Chapter 3 to represent the terminal with its service strategies for trucks, trains, barges and vessels. Experiments, conducted on actual data, illustrated how to use optimization model to determine the number of straddle carriers needed for the next day and to allocate these straddle carriers to different transport modes.

Since stochastic aspects of the container terminal were not included in the optimization model, we used simulation to evaluate the performance of the allocation proposed by the optimization model in a stochastic environment. We introduced a discrete event simulation representing the analyzed terminal and validated it against the results of the optimization model. We performed a sensitivity analysis to evaluate proposed allocation with different levels of variability for straddle carrier handling times.

Results show that the proposed allocation performs well and that the optimization model predicts resulting delays adequately for 10 and 30% of variability. For 50% variability, delays increase considerably for trucks and vessels and are significantly different from the deterministic case. Straddle carriers are shared among trucks and vessels interact with quay cranes, this amplifies the impacts of variability in straddle carrier handling times. Further analysis of these effects could be helpful to improve the allocation proposed by the optimization model to become more robust.

Chapter 5

SCAP and truck appointment systems

Container terminals use different policies like extended gate hours, truck appointment systems or congestion tolls to decrease congestion at the terminal (Morais and Lord; 2006; Maguire et al.; 2010). The underlying idea is to reduce congestion at peak hour periods by controlling and evening out truck arrivals and to minimize the stochasticity of truck operations in the yard. We focus on terminals using truck appointment systems to limit the number of trucks entering the terminal. Several case studies (e.g., Sgouridis and Angelides; 2002; Morais and Lord; 2006; Srour et al.; 2003; Giuliano and O'Brien; 2007) show that appointment systems have the potential to reduce congestion at the terminal. Terminals obtain a better visibility on the moves per day and may adapt operations inside the terminal. The trucking community benefits from faster turnaround times and a higher productivity.

This chapter analyzes the impacts of a truck appointment system on delays of trucks, trains, barges and vessels at intermodal container terminals using straddle carriers to serve all transport modes. Section 5.1 introduces the problem and summarizes related literature. Section 5.2 adapts the mixed integer linear programming model for the straddle carrier allocation problem to a container terminal using a truck appointment system. Section 5.3 evaluates the impact of a truck appointment system on overall delays via experiments with our optimization and simulation models. Section 5.4 concludes the paper.

5.1 Introduction

Truck appointment systems are reservation systems that limit the number of trucks to be served during a specified time slot. Morais and Lord (2006) and Giuliano and O'Brien (2007) report different ways container terminals implement truck appointment systems. Some terminals make the use of the appointment system mandatory, others serve trucks with and without appointments. Terminals have to decide whether to offer appointments on a container or on a truck basis. In the first case, appointments have to be made to deliver or pick up a specific container; in the second case, appointments are made for trucks without further information on container delivery or pick-up. Terminals also differ with regard to the used appointment system provider, the reservation policy (how and when) and the way no-shows are handled. Some terminals prepare container pick-ups based on the made appointments, some make special arrangements at gates for trucks with appointments and others do not differentiate between trucks with and without appointment systems.

These studies also highlight the fact that the success of an appointment system relies on a large percentage of trucks using it. For an insufficient number of participants, the average

truck turnaround time may increase since the service quality of trucks without appointments decreases. This makes it difficult to promote the system as benefits are not seen immediately. Therefore, incentives and clear benefits have to be provided to truck drivers to win them over. Appointment systems have to be flexible (e.g., reservation, cancellation, reassignment of slots) in order to be successful.

5.1.1 Problem description

We combine the straddle carrier allocation problem with the sizing of a truck appointment system. In this case, the number of truck appointments that can be accepted are limited by the number of straddle carriers allocated to trucks. The number of appointments also influences the service quality of trains, barges and vessels since straddle carriers allocated to trucks cannot serve trains, barges or vessels. Consequently, the allocation of straddle carriers and the sizing of the appointment system should be planned simultaneously to minimize overall delays at the terminal. Our objective is to determine a resource allocation proposing appointments close to the preferred arrival pattern of trucks while minimizing the delays of trains, barges and vessels.

We chose to represent container terminals using obligatory appointment systems. Trucks have to book an appointment for a specific container for a specific time slot to enter the terminal to deliver or pick up this container. Entering the terminal without an appointment is not possible. In return and to minimize truck waiting times, each truck is served within a guaranteed service time. To respect preferred truck arrivals, the maximal deviation Δ between preferred and assigned time slots is limited.

5.1.2 Literature review

Several studies determine the number of truck appointments to accept. These studies assume a given capacity (e.g., number of yard cranes) to serve trucks and determine the number of trucks that may be accepted with this capacity to obtain the desired service quality. These studies neglect interactions with other transport modes such as trains, barges and vessels.

Chen et al. (2011) develop a convex nonlinear programming model to determine the number of appointments based on an analytical point-wise stationary approximation model to represent time-dependent truck queuing processes at the gate and in the yard. The objective is to minimize the total truck turnaround time and the difference between preferred and assigned arrival times. They also propose a method to determine time-varying tolls that lead to the optimized truck arrival pattern. Ioannou et al. (2006) develop an algorithm to generate cooperative time windows: container terminals generate wide time windows and trucking companies choose a narrower time window (of predetermined width) within the original ones or renegotiate the time windows offered by the container terminal. To determine the narrower time windows, they propose a heuristic based on an insertion method to solve the Traveling Salesman Problem with Time Windows. Murty et al. (2005) present a decision support system developed for the Hong Kong International Terminals which includes a simulation model to determine the number of appointments to accept. They minimize a combined penalty for yard-crane idle time and the fraction of time during which the queue of trucks waiting at the block for service from the yard crane are too long.

Chen, Govindan and Yang (2013) focus on container terminals assigning time windows to each vessel for container deliveries and pick-ups to shorten container storage time. They introduce a method to optimize these time windows to reduce truck waiting time, idling fuel consumption, cargo storage time and storage yard fee. They estimate the length of the truck queue with a non-stationary queuing model. Time window optimization is done by different variants of genetic algorithms. Huynh and Walton (2008) examine the effect of limiting truck arrivals on truck turn time and crane utilization via a combination of mathematical formulation and simulation. They show that limiting truck arrivals can be beneficial. But, setting caps to low is to the detriment of truckers and the terminal since resources remain partly unused. Their methodology can also be used to determine the optimal number of trucks to accept per time slot. Guan and Liu (2009) present a multi-server queuing model to analyze gate congestion and to quantify the truck waiting cost. Their study is limited to the entry gates and does not include possible congestion and waiting times within the terminal. A nonlinear optimization problem with discrete variables determines the optimal number of gate lanes to open while minimizing a combined cost of truck waiting times and gate operating costs.

Other studies aim to minimize truck waiting times but do not analyze appointment systems. Huynh (2005) presents several regression models and a simulation model to determine the number of yard cranes needed to achieve a desired service level for trucks. Other authors (e.g., Kim et al.; 2003; Zhao and Goodchild; 2010) use the information on truck arrivals to determine their service sequence in order to minimize the number of parasite movements. Namboothiri (2006) and Namboothiri and Erera (2007) deal with appointment systems from the truckers' point of views. They study methods for managing a drayage fleet serving a port with an appointment-based access control system.

5.2 Mixed integer program

We adapt the aggregated model for the straddle carrier allocation problem to represent container terminals using obligatory truck appointment systems. As before, the truck submodel may be combined with other submodels to represent the entire terminal. Figure 5.1 illustrates the network flow model used to assign truck tasks to time slots and to allocate straddle carriers to trucks. The empty and shaded circle nodes represent the discrete time periods of the working day. Flows p_r^m represent the number of tasks that should be executed in period r. Flows $Z_{r,t}^m$ assign each container to a time slot, respecting the maximum deviation Δ, during which it has to be delivered or picked up. Here, we set $\Delta = 1$. Flows W_t^m represent the total number of tasks to be served per period. This number is limited the capacity of allocated straddle carriers.

We use the following parameters to represent the expected truck workload and the capacity at the terminal:

Parameters:

T	Number of time periods describing the working day
r	Index of a time period related to trucks preferred arrivals, $r = 1, \ldots, T$
t	Index of a time period related to trucks assigned arrivals, $t = 1, \ldots, T$
m	Index representing the transport mode 'truck'

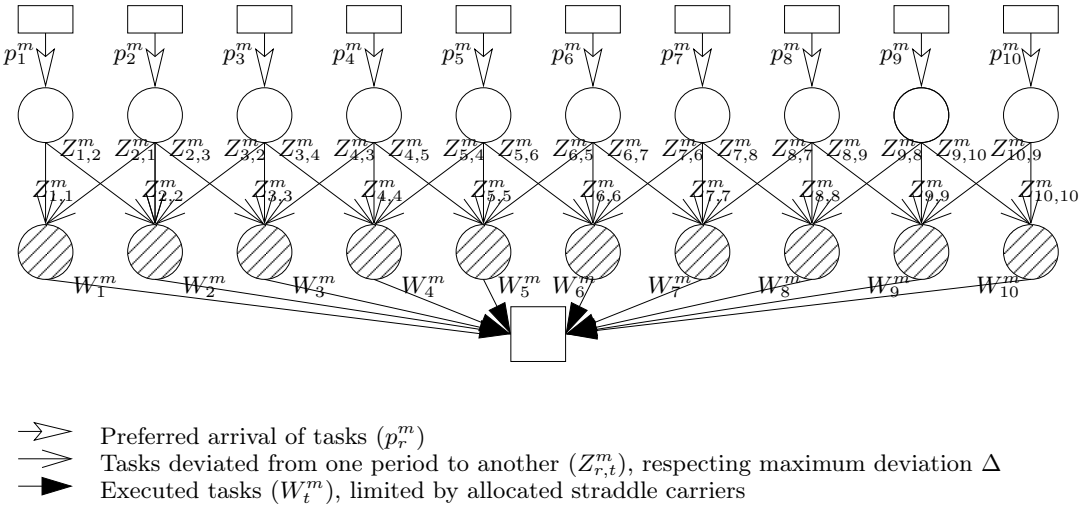


Figure 5.1: Scheme of the network flow model for the dimensioning of a truck appointment system

p_r^m	Aggregated number of tasks wishing to arrive at period r
Δ	Maximal allowed deviation ($\pm$) from preferred arrival time
$w_{r,t}^m$	Cost for deviating a task from period r to period t, $w_{r,t}^m = 0$ if $t = r$ and $w_{r,t}^m > 0$ otherwise
s_t	Number of straddle carriers available at period t
h^m	Average number of tasks a straddle carrier serving transport mode m can handle per period

The model's output indicates the number of truck tasks to be executed per period and an optimal allocation of straddle carriers. It also displays the deviations resulting from this allocation. The following variables represent this information:

Variables:

X_t^m	Number of straddle carriers allocated to trucks in period t
$Z_{r,t}^m$	Number of tasks wishing to arrive in period r that are assigned to period t
W_t^m	Number of tasks assigned to period t indicating the number of appointments to offer

These parameters and variables enable us to formulate the problem.

$$\min \sum_{r=1}^{T} \sum_{t=1}^{T} w_{r,t}^c \cdot Z_{r,t}^m$$

s.t.

$$p_r^m = \sum_{t=1}^{r+\Delta} Z_{r,t}^m \qquad \forall r = 1, \ldots, \Delta \tag{5.1}$$

$$p_r^m = \sum_{t=r-\Delta}^{r+\Delta} Z_{r,t}^m \qquad \forall r = \Delta + 1, \ldots, T - \Delta \tag{5.2}$$

$$p_r^m = \sum_{t=r-\Delta}^{T} Z_{r,t}^m \qquad \forall r = T-\Delta+1,\ldots,T \tag{5.3}$$

$$W_t^m = \sum_{r=1}^{t+\Delta} Z_{r,t}^m \qquad \forall t = 1,\ldots,\Delta \tag{5.4}$$

$$W_t^m = \sum_{r=t-\Delta}^{t+\Delta} Z_{r,t}^m \qquad \forall t = \Delta+1,\ldots,T-\Delta \tag{5.5}$$

$$W_t^m = \sum_{r=t-\Delta}^{T} Z_{r,t}^m \qquad \forall t = T-\Delta+1,\ldots,T \tag{5.6}$$

$$W_t^m \leq h^m \cdot X_t^m \qquad \forall t = 1,\ldots,T \tag{5.7}$$

$$W_t^m \geq h^m \cdot (X_t^m - 1) + 1 \qquad \forall t = 1,\ldots,T \tag{5.8}$$

$$X_t^m \leq s_t \qquad \forall t = 1,\ldots,T \tag{5.9}$$

$$X_t^m \in \mathbb{N}^+ \qquad \forall t = 1,\ldots,T \tag{5.10}$$

$$Z_t^m \in \mathbb{R}^+ \qquad \forall t = 1,\ldots,T \tag{5.11}$$

$$W_{r,t}^m \in \mathbb{R}^+ \qquad \forall r = 1,\ldots,T, t = 1,\ldots,T \tag{5.12}$$

Constraints (5.1) to (5.6) formulate the mass balance constraints for incoming, deviated and executed tasks for trucks. These constraints respect the maximum deviation and impose that all tasks are served within one working day. Constraints (5.7) to (5.10) are identical to the case without an appointment system. Constraint (5.7) limits the number of executed tasks by the allocated capacity. Constraint (5.8) imposes that each allocated straddle carrier executes at least one task. Constraint(5.9) makes sure that the number of allocated straddle carriers does not exceed the number of available straddle carriers. Constraint (5.10) makes it possible to share straddle carriers among trucks and prevents sharing among other transport modes. Constraints (5.11) and (5.12) define variables domains.

5.3 Numerical experiments

We estimates the impacts of a truck appointment system on the service quality of trucks as well as on trains, barges and vessels. We compare the delays at terminals with and without an appointment system. The analysis is done for the container terminal at the Grand Port Maritime de Marseille with our optimization and simulation models.

5.3.1 Results optimization model

To represent the container terminal with a truck appointment system, we combine the adapted truck submodel with the submodels for trains, barges and vessels presented in Section 4.2. We solve the model for the instances presented in Section 4.1.2 for 10, 12 and 14 available straddle carriers and maximum deviations of 1 and 2 periods.

Table 5.1 shows the delays for trucks, trains and barges for terminals with and without appointment systems. For both models, delays for trains are indicated by the number of non-executed tasks and delays for barges by the number of periods spent in the terminal. With an appointment system, truck delays represent the deviation between the preferred and the assigned arrival of tasks; without an appointment system,truck delays represent periods spent at the terminal. Vessels may not be delayed and are not displayed. Since we aim to serve trucks at their preferred arrival times, offered truck appointments are identical to preferred truck arrivals when enough resources are available to do so. In this case, results with and without appointment systems are identical. Instances with different delays for the different terminals are marked in bold.

With 14 available straddle carriers almost no delays occur at the terminal: very few truck tasks are delayed, all train tasks are executed and all barges are served within the minimum service time (resulting from quay crane capacity). If the number of available straddle carriers is reduced to 12 or 10, more delays occur. Some instances may even become infeasible which means that the number of available straddle carriers is not sufficient to serve all tasks within their imposed time windows. Without an appointment system, trucks can be delayed until the end of the working day. With an appointment system, the arrival of trucks can only be forwarded or postponed by maximal Δ periods and trucks have to be served within the period to which they are assigned. This explains why instance 9_10SC is only infeasible for $\Delta = 1$.

Without an appointment system, truck tasks can only be delayed from one period to the next until the end of the working day; with an appointment system, tasks can be forwarded and delayed by maximal Δ periods. It appears that the additional possibility to forward tasks reduces truck delays considerably for most instances. Increasing the maximum deviation from $\Delta = 1$ to $\Delta = 2$, increases room for maneuver and reduces truck delays further. These results suggest that the introduction of a truck appointment system reduces the time trucks have to spend at the terminal, especially if the workload is excessive. In addition, deviations from the preferred arrival are probably less disturbing for trucks than long service times at the terminal.

For instances 9_12SC and 10_12SC truck delays increase with the introduction of an appointment system or an extension of Δ. In both cases, these additional delays are made up by the reduction of delays of barges or trains. For instances 2_10SC and 9_10SC, trucks and barges benefit from the introduction of the appointment system. For instance 9_10SC however, delays for trains increase with the introduction of the truck appointment system. In this case, trains are delayed as the instance would be infeasible otherwise. These results suggest that trains and barges may also benefit from the truck appointment system: trucks may be shifted to less busy periods and free straddle carriers for barges and trains. They also highlight the importance of choosing the right values for weights w^m and the maximum deviation Δ to represent priorities among transport modes correctly.

Our objective is to offer appointments close to the preferred arrival of tasks. Figure 5.2 compares the average arrival of truck tasks per period with the average number of appointments offered per period for a maximum deviation of one and two periods. Only instances where truck arrivals are changed are included. The number of proposed appointments is close to the preferred arrival pattern. With an appointment system, fewer tasks arrive for periods 4 to 6. In fact, tasks are forwarded and postponed to free resources for barges which arrive at period 4. We also observe that the peak arrival at period 10 is smoothed out.

Table 5.1: Delays of trucks, trains and barges at the terminal without an appointment system (w/o) and with a truck appointment system with a maximum deviation of one ($\Delta = 1$) and two ($\Delta = 2$) for 14, 12 and 10 available straddle carriers (SC)

		1	2	3	4	5	6	7	8	9	10
	Truck - sum of delays / deviations										
14 SC	w/o	0	4	0	0	0	0	0	4	1	0
	$\Delta = 1$	0	4	0	0	0	0	0	4	1	0
	$\Delta = 2$	0	4	0	0	0	0	0	4	1	0
12 SC	w/o	0	**55**	1	0	0	0	0	inf	**63**	**inf**
	$\Delta = 1$	0	**43**	1	0	0	0	0	inf	**102**	**132**
	$\Delta = 2$	0	**43**	1	0	0	0	0	inf	**102**	**190**
10 SC	w/o	0	**275**	**247**	**26**	0	**115**	**129**	inf	**607**	inf
	$\Delta = 1$	0	**214**	**165**	**25**	0	**48**	**66**	inf	**inf**	inf
	$\Delta = 2$	0	**204**	**139**	**25**	0	**48**	**66**	inf	**598**	inf
	Train - number of non-executed tasks										
14 SC	w/o	0	0	-	0	0	0	0	0	0	0
	$\Delta = 1$	0	0	-	0	0	0	0	0	0	0
	$\Delta = 2$	0	0	-	0	0	0	0	0	0	0
12 SC	w/o	0	0	-	0	0	0	0	inf	0	**inf**
	$\Delta = 1$	0	0	-	0	0	0	0	inf	0	**37**
	$\Delta = 2$	0	0	-	0	0	0	0	inf	0	**23**
10 SC	w/o	0	1	-	0	0	0	0	inf	**2**	inf
	$\Delta = 1$	0	1	-	0	0	0	0	inf	**inf**	inf
	$\Delta = 2$	0	1	-	0	0	0	0	inf	**9**	inf
	Barge - number of periods spent at the terminal										
14 SC	w/o	-	4	-	6	4	-	2	-	3	-
	$\Delta = 1$	-	4	-	6	4	-	2	-	3	-
	$\Delta = 2$	-	4	-	6	4	-	2	-	3	-
12 SC	w/o	-	4	-	6	4	-	2	inf	**4**	inf
	$\Delta = 1$	-	4	-	6	4	-	2	inf	**3**	-
	$\Delta = 2$	-	4	-	6	4	-	2	inf	**3**	-
10 SC	w/o	-	**6**	-	6	4	-	2	inf	**5**	inf
	$\Delta = 1$	-	**4**	-	6	4	-	2	inf	**inf**	inf
	$\Delta = 2$	-	**4**	-	6	4	-	2	inf	**3**	inf

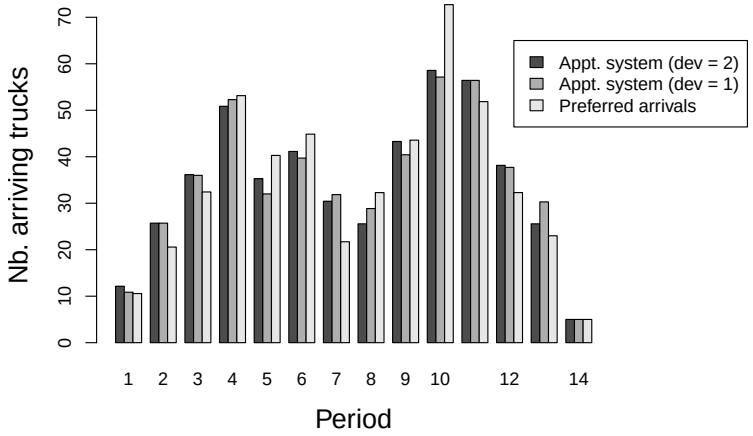


Figure 5.2: Preferred truck arrivals and proposed appointments

5.3.2 Results simulation model

The optimization model suggests that the use of an appointment system improves the performance of the terminal. We use the simulation model presented in Section 4.3 to validate this finding in a stochastic environment. We compare two cases: a realistic and an ideal appointment system. For the realistic case, arrivals differ slightly from the planned appointments. We model truck arrivals as a non-stationary Poisson process with averages set to the number of appointments obtained by the optimization model. For the ideal case, trucks arrive evenly distributed within the period to which they are assigned. We impose truck arrivals via a given schedule.

We run experiments only for those 8 instances where truck delays differ for terminals with and without an appointment system for a maximum deviation of 2. We analyze three scenarios with $\pm 10\%$, $\pm 30\%$ and $\pm 50\%$ of variability of handling and traveling times of straddle carriers. 1000 replications are executed for each instance. Table 5.2 shows the results for cases without, with a realistic and with an ideal appointment system. It indicates the average truck service time in minutes and the standard deviation, as well as average delays of trains, barges and vessel.

The realistic appointment system reduces the average truck service time by approximately 14 minutes. Results of Student's t-tests show that differences are statistically significant for all variability levels with p-values of 0.044, 0.048 and 0.046, respectively. The standard deviation is reduced by 50 to 60 percent. Truck arrivals are stochastic and the benefit results from the possibility to forward or postpone truck arrivals to less busy periods.

In the ideal case, average truck service times decrease by 17 to 22 minutes. Results of Student's t-tests show that differences are statistically significant for all variability levels with p-values of 0.015, 0.012 and 0.008, respectively. The standard deviation is reduced by 70 percent. The additional benefit comes from the fact that truck arrivals equal the number of offered appointments and are evenly distributed over a time period; allocated resources are used very efficiently.

Table 5.2: Delays for terminals without, with realistic and with ideal truck appointment systems for different levels of variability of straddle carriers traveling and handling times

	10% variability			30% variability			50% variability		
	none	realistic	ideal	none	realistic	ideal	none	realistic	ideal
Truck service time [min]	32.86	19.40	15.64	34.82	21.00	16.00	40.33	25.57	18.30
Std dev.	23.69	10.72	8.38	24.87	11.72	8.51	27.93	14.38	9.48
Train not served [cont]	0.21	0.96	0.96	0.21	0.96	0.96	0.35	1.07	1.09
Std dev.	0.16	0.22	0.22	0.23	0.25	0.26	0.41	0.34	0.40
Barge not served [cont]	0.00	0.01	0.07	0.01	0.05	0.13	0.24	0.39	0.68
Std dev.	0.03	0.09	0.20	0.12	0.23	0.37	0.66	0.90	1.16
Vessels not served [cont]	2.92	3.17	3.49	6.22	6.53	6.96	17.63	18.03	18.63
Std dev.	4.27	5.12	5.35	6.11	7.23	7.42	8.46	9.86	9.92

For the realistic and ideal appointment system, the allocation of straddle carriers to trains, barges and vessels is identical. Results differ slightly due to stochastic inputs. We concentrate our analysis on the differences between the terminal without an appointment system and the realistic case. The average train delay increases if the appointment system is used. This results from the allocation plan proposed by the optimization which results in 10 non-executed tasks with an appointment system against 3 non-executed tasks without an appointment system. Results of Student's t-tests show that differences are not statistically significant. The corresponding p-values are 0.455, 0.453 and 0.492, respectively.

Delays for barges are very similar for all cases, but seem to increase slightly if the appointment system is used. With an appointment system every barge is served as soon as it enters the terminal. Due to stochastic arrivals barges may arrive delayed and straddle carriers may remain unused at the beginning of the period. If the service of barges starts at a later period, like it is the case for three instances without an appointment system, variability of arrivals do not result in unused capacity. Results of Student's t-tests show that differences are not statistically significant. The corresponding p-values are 0.261, 0.191 and 0.165, respectively.

For vessels, delays are very similar for all cases. Differences result from different allocations: the numbers of allocated straddle carriers are identical, but their distributions over time differ. Results of Student's t-tests show that differences are not statistically significant. The corresponding p-values are 0.485, 0.771 and 0.813, respectively.

5.4 Conclusion

This chapter analyzed the impacts of a truck appointment system on delays of trucks as well as trains, barges and vessels. We combined the straddle carrier allocation problem with the

dimensioning of the truck appointment system since the number of appointments to offer depends on the number of allocated straddle carriers.

We adapted the mixed integer model for the straddle carrier allocation problem to represent terminals using obligatory truck appointment systems. The objective is to simultaneously determine the number of truck appointments to offer and a straddle carrier allocation to reduce overall delays at the terminal.

Experiments conducted via optimization and simulation for the container terminal at the Grand Port Maritime de Marseille indicate that the truck appointment system may reduce truck delays and also delays of trains, barges and vessels. The appointment system may deviate truck arrivals to free resource for the other transport modes. Benefits for trucks may be even higher than suggested by our comparison since they are aware of deviations a priori and do not need to wait at the terminal.

Simulation results also show that precise information on truck arrivals reduces truck delays and that the distribution of allocated straddle carriers over time has impacts on delays. It may thus be beneficial to get some more insight on the impacts of the allocation pattern on delays and to include these findings in the optimization model.

PART II

Container relocation problem

Container terminals stack containers to use their scarce land efficiently. They use GPS or RFID to keep track of exact container locations. The drawback of stacking is that only the topmost container of each stack can be accessed directly. If another container has to be retrieved, parasite movements are necessary to relocate blocking containers. Poor yard management increases the number of relocations and the time needed to retrieve containers. Thereby, it decreases the overall productivity of the terminal. Relocations cannot be avoided completely as little information about future retrievals is known when a container has to be stored.

This part presents our work on the container relocation problem where a set of containers has to be retrieved in a given sequence with a minimum number of relocations. Existing studies on the container relocation problem assume that the entire retrieval sequence is known in advance. But, in reality container terminals have only limited information on future retrievals, especially for containers picked up by trucks. Our final objective is to address a dynamic version of the container relocation problem where information on future retrievals is revealed over time. We first studied the static version to get a better understanding of the problem. We then started working on the dynamic version, but many possibilities exist to continue this work.

Chapter 6 introduces the container relocation problem (CRP), summarizes related studies and introduces bounds on the number of relocations. Chapter 7 presents and improves an existing binary programming model. This model solves small and medium instances, but is impractical for bigger instances. Chapter 8 presents an exact branch and price approach. It formulates the master problem based on the binary model of the previous chapter. It also describes different pricing subproblems (based on mixed integer programming and enumeration) and the branching procedure. But, the exact subproblems do not generate columns quickly. Chapter 9 presents a heuristic branch and price approach. It uses column generation with a heuristic subproblem and runs a heuristic repeatedly. Its objective is to obtain good integer solutions rather than the optimal fractional solution. Chapter 10 deals with a dynamic version of the problem where the retrieval sequence is revealed over time. It presents and compares different relocation strategies for the dynamic case.

Chapter 6

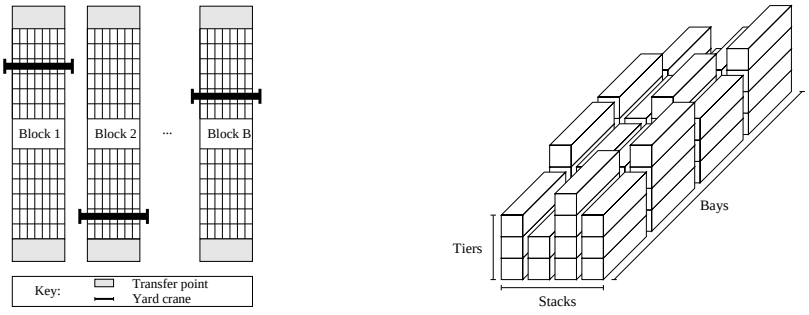
Container relocation problem (CRP)

This chapter introduces the container relocation problem. Section 6.1 gives some background on yard optimization at container terminals and states the container relocation problem as dealt with in academic literature. Section 6.2 summarizes related literature. Section 6.3 presents existing upper and lower bounds on the number of relocations and introduces a new upper bound. Section 6.4 details instances that are used for experiments throughout the next chapters.

6.1 Problem description

To decouple seaside and landside operations, incoming containers are not immediately loaded on an outgoing vehicle, but stored in the yard for up to several days. Due to limited space, terminals stack containers. Consequently, only the topmost container of each stack can be accessed directly. If another container has to be retrieved, containers above have to be relocated. These unproductive relocations (also called reshuffles or rehandles) should be avoided since they increase the retrieval time and hence the overall performance of the terminal. However, relocations cannot be avoided completely as little information about future retrievals is known when a container has to be stored.

The number of relocations increases with the stacking height of containers and is therefore a bigger issue at terminals using stacking cranes for storage operations. The yard of such a terminal is illustrated in Figure 6.1. The yard is divided into different blocks. Each block consists of several bays, each bay of several stacks and each stack of several tiers. Thanks to new technologies, the terminal knows exactly at which position (block, bay, stack, tier) each container is stored and which positions are empty.



(a) A container yard (b) A single block

Figure 6.1: Blocks, bays, stacks and tiers

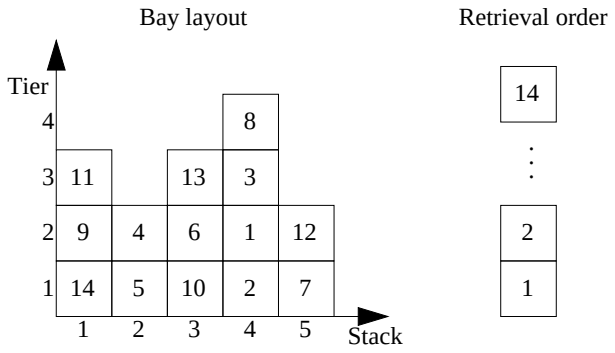


Figure 6.2: Container relocation problem

Decisions where to place containers are taken, when containers enter the terminal or when they have to be relocated. Different academic problems have been extracted for yard optimization: the storage space allocation problem to determine storage locations (a block or a single position) for incoming containers; the remarshalling / premarshalling problem to reorganize a block / a bay in less busy periods as new information becomes available in order to reduce the number of relocations during the retrieval process; the container relocation problem to retrieve all containers from a bay in a given sequence with a minimum number of relocations.

We deal with the container relocation problem. In this case, the stowage plan of vessels and the service order of trucks are known and impose the retrieval order of containers. Generally, the storage layout does not match the retrieval order and containers have to be relocated. Figure 6.2 illustrates the problem. The objective is to retrieve all containers in the given sequence with a minimum number of relocations. Two variants of the problem exist: all containers may be relocated or only containers above the current target container may be relocated. The problem definition relies on assumptions A1 to A7 and may or may not include assumption A8.

A1: The initial bay layout and precedence constraints among single containers or groups of containers are known in advance.

A2: No new containers arrive during the retrieval process.

A3: Only the topmost container of a stack can be picked up. A relocated container can only be put on the top of another stack or on the ground.

A4: Containers are only relocated within the bay since relocations between bays are very time consuming.

A5: The bay size is limited by the maximum numbers of stacks and tiers.

A6: Containers in the same bay have the same size and can be piled up in any order.

A7: The distance traveled within one bay (horizontally and vertically) has little impact on the time to relocate or to retrieve containers.

A8: Only blocking containers located above the current target container may be relocated.

Like most other studies, we address the container relocation problem with precedence constraints among single containers and relocate only containers above the target container (A8). We call these containers blocking containers. We use the notation introduced by Caserta et al. (2012) to represent the container relocation problem. A bay consists of W stacks and H tiers. Each slot within the bay is addressed with coordinates (i, j) where $i \in \{1, \ldots, W\}$ and $j \in \{1, \ldots, H\}$. The initial configuration contains N containers, labeled $1, \ldots, N$. Containers have to be retrieved in ascending order, e.g. container 1 is the first one to be retrieved and container N the last one. At each time period t ($t = 1, \ldots, T$), container $n = t$ is retrieved and any blocking containers are relocated.

6.2 Related literature

We provide an overview of studies dealing with the container relocation problem. The problem is NP-hard (Caserta et al. (2012)) and few exact and several heuristic solution approaches exist. For a broader review of literature on yard optimization refer to Caserta, Schwarze and Voß (2011).

Lee and Hsu (2007) present different integer linear programming models based on a multi-commodity flow problem with a set of side constraints for several stacking problems including the container relocation problem. Caserta et al. (2012) present two binary programming models for the problem with and without assumption A8. The models contain state variables to represent the bay layout and movement variables to represent retrievals and relocations of containers. They also propose a simple heuristic based on the computation of a stack score. Petering and Hussein (2013) introduce a formulation with fewer decision variables for the problem without assumption A8. Binary variables indicate the stack in which a container is in and real variables its relative position to the top of this stack. They also adapt the heuristic above to the case without assumption A8. Tang et al. (2012) deal with the plate shuffling problem which is identical to the container relocation problem except that steel plates are relocated instead of containers. Binary variables indicate the stack in which a container is in, if two plates are in the same stack and if one plate is above another one. They minimize the number of relocations and the traveled crane distance. They present a tabu search to solve the problem.

Kim and Hong (2006) study the problem with priorities among single containers and among groups of containers. They use a branch and bound algorithm that branches over all bay layouts that may be reached by retrieving one container and relocating containers above. They determine the expected number of additional relocations based on probabilities of container relocations. They propose a heuristic relocating containers based on this value. Wu and Ting (2010) apply different branch and bound based heuristics using different simple relocation strategies. Ünlüyurt and Aydin (2012) aim to minimize the number of relocations and the horizontally traveled distance. They apply a branch and bound approach and existing heuristics to solve the problem.

Zhang et al. (2010) present an iterative deepening A* algorithm (IDA*) and introduce lower bound measures. Zhu et al. (2012) extend IDA* to the case without assumption A8. Forster and Bortfeldt (2012) present a tree search procedure for the problem without assumption A8. Their approach is based on a classification of feasible moves, a greedy approach to find an initial solution, a lower bound for the number of moves and a branching scheme.

Rei and Pedroso (2012) study the scalability of an exact approach to this problem and propose two heuristic methods: a multiple simulation algorithm using semi-greedy construction heuristics and a stochastic best-first tree search algorithm.

Wu et al. (2009) use tabu-search with a one-dimensional vector to represent relocations and retrieval operations. Two simple heuristics are used to generate the initial solution and two move operators are applied to obtain neighborhood solutions. Caserta, Voß and Sniedovich (2011) use a corridor method which reduces the exponentially large number of possible states by adding exogenous constraints to limit the number of possible relocations. A dynamic programming scheme is then applied to the restricted solution space.

Caserta et al. (2009) and Caserta and Voß (2009) present two random based procedures. For each container to be relocated, they define a neighborhood representing reachable bay configurations. A greedy score is calculated for each configuration and a roulette-wheel mechanism selects the next configuration with its associated move. Caserta et al. (2009) propose a binary description of the problem and apply this random based procedure. Caserta and Voß (2009) use the corridor method to define the neighborhood of the current configuration within the random based procedure.

6.3 Bounds on the number of relocations

This section presents existing upper and lower bounds on the number of relocations needed to clear the bay. The upper bound (UB) is obtained via a heuristic and the lower bound (LB) from the initial bay layout. Based on these bounds, we introduce an optimality criterion and a new upper bound ($R^t_{\max,1}$) limiting the maximum number of relocations per period.

Upper bound on the number of relocations The heuristic introduced by Caserta et al. (2012) uses stack scores $s(i)$ for $i = 1$ to W to decide into which stack i^* a container should be relocated to. Score $s(i)$ indicates the first container that has to be retrieved from stack i. Scores are computed according to Equation (6.1).

$$s(i) = \begin{cases} \min(n) \mid n \text{ placed in stack } i & \text{if at least one container is in stack } i, \\ N+1 & \text{if stack } i \text{ is empty.} \end{cases} \tag{6.1}$$

The heuristic chooses stack i^* where container n should be relocated to according to Equation (6.2). It prefers to put container n into a stack with $s(i) > n$. This does not cause an additional relocation since container n has to be retrieved prior to all other containers in stack i. If several stacks with $s(i) > n$ exist, the container is relocated to the stack with the smallest $s(i) > n$ since stacks with large $s(i)$ are valuable. If no stack i with $s(i) > n$ exists, the container causes an additional relocation in subsequent periods. It is put into stack i with maximum $s(i)$ to be relocated again as late as possible. Their heuristic makes sure that containers are not relocated into the stack from which they have been retrieved. But, it seems to neglect the fact that some stacks may already be full and cannot receive containers. We introduce set $\mathcal{W} \subset \{1, \ldots, W\}$ to describe the set of stacks where containers can be relocated to. It contains only stacks other than the retrieval stack that have at least one empty position.

$$i^* = \begin{cases} \text{argmin}_{i \in \mathcal{W}} \{s(i) | s(i) > n\} & \text{if } \exists i \text{ with } s(i) > n, \\ \text{argmax}_{i \in \mathcal{W}} \{s(i)\} & \text{otherwise.} \end{cases} \tag{6.2}$$

Algorithm 6.1 presents the complete heuristic. In the sequel, we refer to this heuristic as Heuristic HC. At each period t, only containers above the target container may be relocated. The heuristic relocates these containers starting from the topmost container to the container just above the target container. This ensures that the LIFO order is respected. It relocates every relocation container n to its target stack i^* and updates the stack score $s(i^*)$ and the set $\mathcal{W}$. If the current target container is accessible, it is retrieved and the stack score updated. The heuristic stops if all containers are retrieved and returns the number of executed relocations.

Algorithm 6.1 Heuristic HC for the container relocation problem

INPUT: a bay layout
OUTPUT: a solution for the container relocation problem
nb_relocations $\leftarrow 0$
for $i = 1$ **to** W **do**
determine $s(i)$ using Equation (6.1)
end for
for $t = 1$ **to** T **do**
determine $\mathcal{W}$
while $\exists$ container above target container **do**
n $\leftarrow$ topmost relocation container
determine i^* for n using Equation (6.2)
relocate container n to stack i^*
nb_relocations $\leftarrow$ nb_relocations +1
$s(i^*) \leftarrow \min\{s(i^*), n\}$
update $\mathcal{W}$
end while
retrieve target container from s'
determine $s(i')$ using Equation (6.1)
end for
return nb_relocations and executed retrievals and relocations

Figure 6.3 applies Heuristic HC to an example with 6 containers. Figures 6.3a to 6.3f present bay layouts at periods $t = 1$ to $t = 6$. They also indicate retrievals and relocations executed by Heuristic HC. The heuristic executes 5 relocations to retrieve containers 1 to 6 from the bay.

Zhu et al. (2012) improve Heuristic HC by introducing an additional criterion. It applies if the following three conditions are fulfilled: at least two containers n_1 and n_2 have to be relocated, exactly one stack i exists with $n_1 < s(i) < n_2$ that has exactly one empty tier, and $s(j) < n_1$ for all stacks $j \neq i$. If container n_2 has to be relocated first, Heuristic HC relocates it to stack i. Stack i is then full and container n_1 has to be relocated to a stack s_j with $s_j < n_1$. This causes an additional relocation in a later period. The improved heuristic keeps stack $s(i)$ free for container n_1 and relocates container n_2 to stack $j \neq i$ with maximal

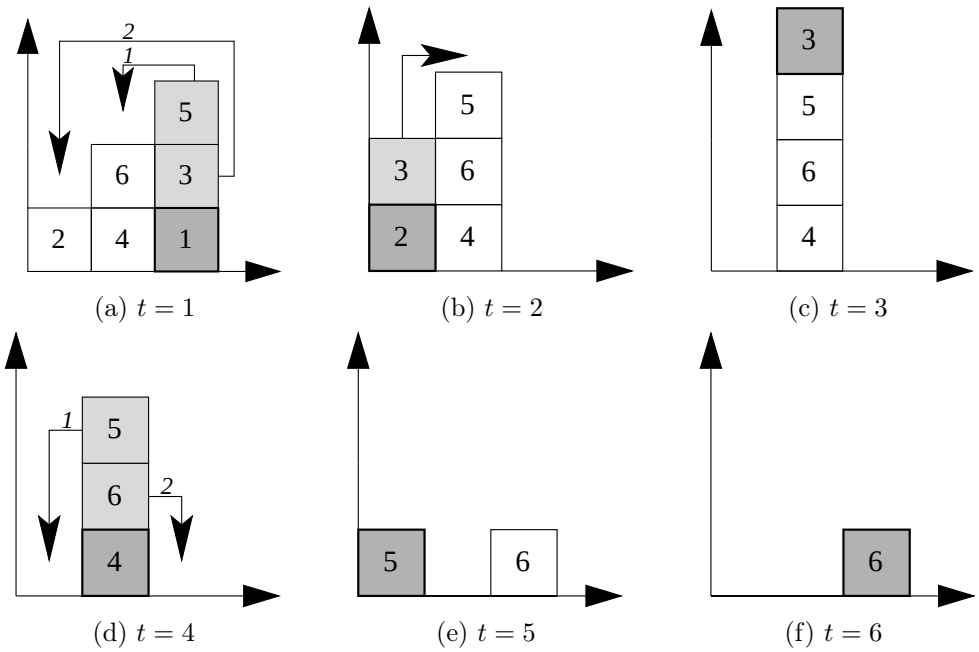


Figure 6.3: Application of Heuristic HC to an example (5 relocations)

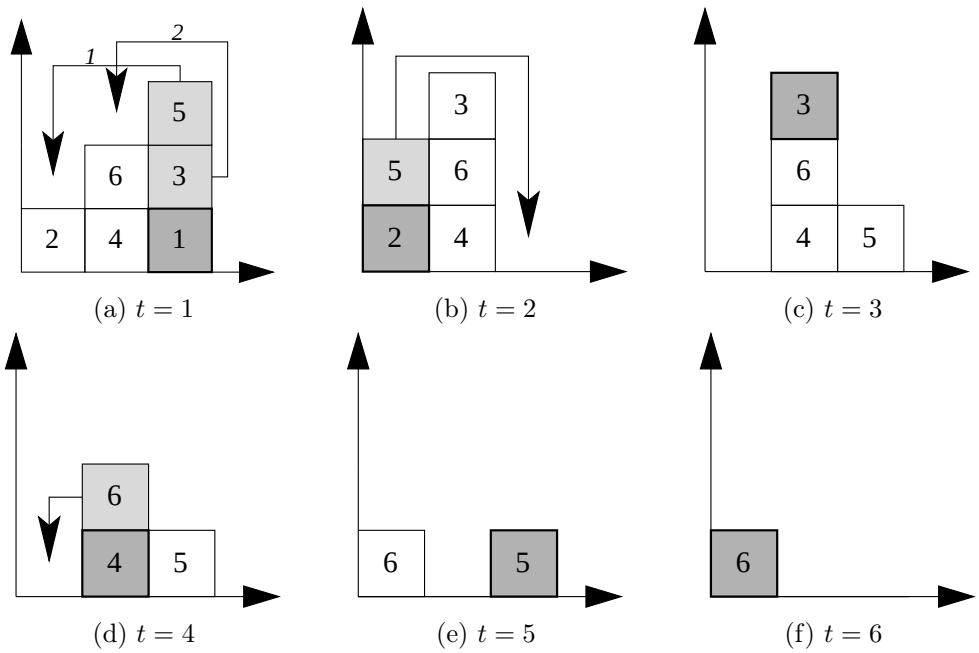


Figure 6.4: Application of the improved heuristic to the same example (4 relocations)

$s(j)$. Container n_1 can then be relocated to stack i with $s(i) < n_1$ and does not cause additional relocations in later periods.

Figure 6.4 illustrates the benefit on the previous example. The improved heuristic relocates containers 3 and 5 differently at period 1 and applies the same relocation rules for periods 2 to 6. Container 3 is never relocated and the heuristic needs only 4 relocation - one less than Heuristic HC - to retrieve containers 1 to 6 from the bay. Zhu et al. (2012) compared the performance of both heuristics on 12 500 instances with different bay sizes. The average number of relocations over all instances is reduced by around 1% from 33.36 to 33.07.

Lower bound on the number of relocations Zhang et al. (2010) introduced lower bound LB. It is computed from the initial bay layout with the help of LB_t and LB_{t+}. LB_t represents the minimum number of relocations needed to retrieve container t. It counts how many containers $n > t$ are located above container t in the initial bay layout. LB_{t+} represents the minimum number of additional relocations caused in subsequent periods by relocations in period t. It counts how many containers n have to be relocated to a stack containing containers $n' < n$. Once a container is relocated, we have no information on its position and it cannot be considered in subsequent periods. LB equals the sum of LB_t and LB_{t+} over all periods t. Algorithm 6.2 illustrates how to compute LB from the initial bay layout.

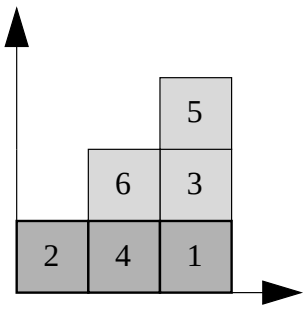
Algorithm 6.2 Determine LB via LB_t and LB_{t+}

INPUT: a bay layout
OUTPUT: a lower bound LB on the number of relocations

```
LB ← 0
for t = 1 to T do
    LB_t ← number of containers placed above container t
    LB_t+ ← number of additional relocations caused by relocating containers n
            to stacks with ∃ n' < n
    delete container t and its blocking containers from the bay
    LB ← LB + LB_t + LB_t+
end for
return LB
```

Figure 6.5 illustrates the computation of LB_t, LB_{t+} and LB. To retrieve container 1, containers 3 and 5 have to be relocated and we obtain $L_1 = 2$. Container 5 can be relocated to stack 1 above container 2 or to stack 2 above containers 4 and 6. In both cases, it has to be relocated again when container 2 or 4 is retrieved. Container 2 can be put on stack 2 without causing additional relocations since it is retrieved before containers 4 and 6. We obtain $LB_{1+} = 1$. No containers are placed above container 2 and we set $LB_2 = 0$ and $LB_{2+} = 0$. Container 3 is already deleted from the bay due to its relocation in $t = 1$ and we set $LB_3 = 0$ and $LB_{3+} = 0$. The values for the remaining LB_t and LB_{t+} are calculated in the same way. We obtain $LB = 4$ which indicates that at least 4 relocations are necessary to retrieve all containers from the bay.

Tang et al. (2012) improve this lower bound by including the number of empty tiers. When computing LB_{t+}, we only checked if at least one stack with $s(i) > n$ exists. They



	Direct relocations (LB_t)	Indirect relocations (LB_{t+})
$t = 1$	2	1
$t = 2$	0	0
$t = 3$	0	0
$t = 4$	1	0
$t = 5$	0	0
$t = 6$	0	0

$LB = \sum_{t=1}^{6}(LB_t + LB_{t+}) = 4$

Figure 6.5: Computation of lower bounds LB_t, LB_{t+} and LB

check if the number of empty positions in stacks i with $s(i) > n$ is sufficient to take all relocations containers $n' > n$. If not, LB_{t+} is adapted accordingly.

Optimality criterion If Constraint (6.3) holds, the lower bound equals the upper bound and the heuristic solution UB is optimal.

$$UB = \sum_{t=1}^{T} LB_t + \sum_{t=1}^{T} LB_{t+} \tag{6.3}$$

Upper bound on the number of relocations per period We introduce a new upper bound $R^t_{\max,1}$. It is used to improve an existing upper bound on the global number of relocations. In this case, $R^t_{\max,1}$ represents the maximum number of relocations per period t for which a solution with UB - 1 relocations may exist. It is defined by Equation (6.4).

$$R^t_{\max,1} = UB - 1 - \sum_{k=1}^{t-1} LB_k - \sum_{k=t+1}^{T} LB_k - \sum_{k=t}^{T} LB_{k+} \quad \forall t = 1, \ldots, T \tag{6.4}$$

Proof. For each instance, we can compute a global upper bound UB (e.g., via Heuristic HC) and lower bounds LB_t and LB_{t+} for each period $t = 1$ to T. We know that UB = LB + K with $K \geq 0$. If a solution with UB - 1 relocations exists, UB - 1 = LB + K' with $K' = K - 1$ and $K' \geq 0$ holds. Hence,

$$UB - 1 = \sum_{k=1}^{T} LB_k + \sum_{k=1}^{T} LB_{k+} + K' \;\Leftrightarrow\; K' = UB - 1 - \sum_{k=1}^{T} LB_k - \sum_{k=1}^{T} LB_{k+} \tag{6.5}$$

$R^t_{\max,1}$ defines an upper bound on the number of relocations in period t. The following relocations may be executed in period t: sure relocations LB_t, indirect relocations $LB_{t'+}$ of earlier periods and these K' relocations on which we have no further information. This defines $R^t_{\max,1}$ as follows

$$R^t_{\max,1} = LB_t + \sum_{k=1}^{t-1} LB_{k+} + K' \qquad \forall t = 1, \ldots, T \tag{6.6}$$

Combining Equations (6.5) and (6.6) leads to Equation (6.4).

$$\begin{aligned} R^t_{\max,1} &= LB_t + \sum_{k=1}^{t-1} LB_{k+} + \left(UB - 1 - \sum_{k=1}^{T} LB_k - \sum_{k=1}^{T} LB_{k+} \right) \\ &= UB - 1 + LB_t - \sum_{k=1}^{T} LB_k + \sum_{k=1}^{t-1} LB_{k+} - \sum_{k=1}^{T} LB_{k+} \\ &= UB - 1 - \sum_{k=1}^{t-1} LB_k - \sum_{k=t+1}^{T} LB_k - \sum_{k=t}^{T} LB_{k+} \quad \forall t = 1, \ldots, T \end{aligned}$$

□

We illustrate the computation of $R^t_{\max,1}$ on the previous example. Heuristic HC obtained a solution with 5 relocations and we set UB = 5. Values of lower bounds LB_t and LB_{t+} are reported in Figure 6.5. We obtain the following values:

- $R^1_{\max,1} = 5 - 1 - 0 - 1 - 1 = 2$ for $t = 1$,
- $R^2_{\max,1} = 5 - 1 - 2 - 1 - 0 = 1$ for $t = 2$,
- $R^3_{\max,1} = 5 - 1 - 2 - 1 - 0 = 1$ for $t = 3$,
- $R^4_{\max,1} = 5 - 1 - 2 - 0 - 0 = 2$ for $t = 4$,
- $R^5_{\max,1} = 5 - 1 - 3 - 0 - 0 = 1$ for $t = 5$,
- $R^6_{\max,1} = 5 - 1 - 3 - 0 - 0 = 1$ for $t = 6$.

6.4 Instances

Caserta, Voß and Sniedovich (2011) introduce instances[1] that are commonly used to compare different solution methods for the container relocation problem (e.g., Caserta et al.; 2009; Caserta, Voß and Sniedovich; 2011; Caserta et al.; 2012; Forster and Bortfeldt; 2012; Petering and Hussein; 2013; Zhu et al.; 2012). Each instance specifies the size of the bay - width W x height H - and the initial position of N containers with priorities 1 to N. All stacks contain the same number of containers S and the two topmost positions of every stack are empty $S = H - 2$. Consequently $N = W$ x S containers are placed in the bay. They provide 21 sets of instances for different bay sizes with 40 instances per set. Thus, a total of 840 instances.

Table 6.1 provides some indicators to evaluate the difficulty of these instances. It displays the average values for the lower bound LB and the upper bound UB presented in Section 6.3. It also indicates the average gap between the lower and the upper bound and the number of trivial instances where the upper bound equals the lower bound.

Gaps between lower and upper bounds depend mainly on stack height S and remain similar for different bay widths W. Bounds are tight for small and medium instances: the

[1]Instances can be downloaded from `http://iwi.econ.uni-hamburg.de/IWIWeb/GetDocument.aspx?documentid=1468`

Table 6.1: Information on instances from Caserta et al. (2012)

Set *S-W*	*N*	Avg. LB	Avg. UB	Avg. gap	Nb. trivial
3-3	9	4.7	5.1	0.4	27
3-4	12	5.9	6.3	0.5	26
3-5	15	6.8	7.1	0.3	28
3-6	18	8.3	8.5	0.2	36
3-7	21	9.1	9.3	0.2	33
3-8	24	10.5	10.7	0.3	32
4-4	16	9.4	11.0	1.6	8
4-5	20	12.2	13.6	1.4	11
4-6	24	13.2	14.7	1.5	11
4-7	28	15.2	16.9	1.7	9
5-4	20	13.6	16.8	3.2	4
5-5	25	17.0	21.2	4.2	1
5-6	30	20.1	24.3	4.2	2
5-7	35	22.4	26.3	3.9	5
5-8	40	25.6	29.6	4.0	4
5-9	45	28.5	32.4	3.8	3
5-10	50	31.4	35.5	4.2	2
6-6	36	27.0	35.9	9.0	0
6-10	60	41.5	49.9	8.3	2
10-6	60	56.6	101.3	44.7	0
10-10	100	84.1	139.3	55.2	0

average gap is smaller than 1 for S = 3, smaller than 2 for S = 4 and smaller than 5 for S = 5. It is less tight for bigger instances: smaller than 9 for S = 6 and smaller than 56 for S = 10. For 244 instances, the lower bound equals the upper bound and the solution of Heuristic *HC* is optimal. The heuristic performs especially well for wide and low bays, but the solution quality decreases as the bay gets higher or narrower.

The number of relocations increases with the number of containers in the bay (e.g., 3-3 vs 3-8, 4-4 vs 4-7). For the same number of containers, more relocations occur for narrow and high bays than for wide and low bays (e.g., 3-8 vs 4-6 vs 5-5). For wide bays, it is likely to be able to place relocation containers into stacks containing only containers with lower priorities. For narrow bays, this is less likely. The difficulty of an instance increases with the stack height, the percentage of occupied slots and the number of containers placed above containers with higher priorities; and that it decreases with the width of the bay.

Chapter 7

Binary integer program for CRP

This chapter presents a binary integer programming model for the container relocation problem relocating only blocking containers. Section 7.1 reports the model introduced by Caserta et al. (2012) and discusses two errors in their formulation. Section 7.2 presents our reformulation of their model which corrects the errors and reduces the number of variables. It also presents a preprocessing mechanism to fix several variables and introduces cuts. Section 7.3 presents computational results for the reformulated model with and without preprocessing and cuts. These results illustrate the benefits of the preprocessing step. Section 7.4 concludes the chapter.

7.1 Model from Caserta et al.

Caserta et al. (2012) identify two sets of binary variables: configuration variables and movement variables. Configuration variables b_{ijnt} represent the layout of the bay over time. Parameters b_{ijn1} with $t = 1$ represent the initial bay layout. Movement variables y_{ijnt} and x_{ijklnt} represent container retrievals and relocations. Binary parameters v_{nt} indicate if a container is in the bay or already retrieved.

$$b_{ijnt} = \begin{cases} 1 & \text{if container } n \text{ is at position } (i,j) \text{ at the beginning of period } t, \\ 0 & \text{otherwise;} \end{cases}$$

$$x_{ijklnt} = \begin{cases} 1 & \text{if container } n \text{ is relocated from position } (i,j) \text{ to } (k,l) \text{ in period } t, \\ 0 & \text{otherwise;} \end{cases}$$

$$y_{ijnt} = \begin{cases} 1 & \text{if container } n \text{ is retrieved from position } (i,j) \text{ in period } t, \\ 0 & \text{otherwise;} \end{cases}$$

$$v_{nt} = \begin{cases} 0 & \text{if container } n \text{ is in the bay at the beginning of period } t \ (n \geq t), \\ 1 & \text{otherwise } (n < t). \end{cases}$$

These variables and parameters are used to formulate the problem.

$$\sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{n=1}^{N} \sum_{t=1}^{T} x_{ijklnt}$$

s.t.

$$\sum_{i=1}^{W}\sum_{j=1}^{H} b_{ijnt} + v_{nt} = 1 \quad \forall n = 1,\dots,N, t = 1,\dots,T \tag{7.1}$$

$$\sum_{n=1}^{N} b_{ijnt} \leq 1 \quad \forall i = 1,\dots,W, j = 1,\dots H, t = 1,\dots,T \tag{7.2}$$

$$\sum_{n=1}^{N} b_{ijnt} \geq \sum_{n=1}^{N} b_{ij+1nt} \quad \forall i = 1,\dots,W, j = 1,\dots H-1, t = 1,\dots,T \tag{7.3}$$

$$b_{ijnt+1} = b_{ijnt} + \sum_{k=1}^{W}\sum_{l=1}^{H} x_{klijnt} - \sum_{k=1}^{W}\sum_{l=1}^{H} x_{ijklnt} - y_{ijnt} \quad \forall i = 1,\dots,W, j = 1,\dots H, n = 1,\dots,N, t = 1,\dots,T-1 \tag{7.4}$$

$$v_{nt} = \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{t'=1}^{t-1} y_{ijnt'} \quad \forall n = 1,\dots,N, t = 1,\dots,T \tag{7.5}$$

$$1 - \sum_{n=1}^{N} x_{ijklnt} \geq \sum_{n=1}^{N}\sum_{j'=j+1}^{H}\sum_{l'=l+1}^{H} x_{ij'kl'nt} \quad \forall i = 1,\dots,W, j = 1,\dots H, k = 1,\dots,W, l = 1,\dots H, t = 1,\dots,T-1 \tag{7.6}$$

$$M \cdot \left(1 - \sum_{j=1}^{H} b_{ijtt}\right) \geq \sum_{j=1}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=1}^{N}\left(\sum_{i'=1}^{i-1} x_{i'jklnt} + \sum_{i''=i+1}^{W} x_{i''jklnt}\right) \quad \forall i = 1,\dots,W, t = 1,\dots,T \tag{7.7}$$

$$x_{ijilnt} = 0 \quad \forall i = 1,\dots,W, j = 1,\dots H, l = 1,\dots H, n = 1,\dots,N, t = 1,\dots,T \tag{7.8}$$

$$x_{ijklnt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 1,\dots H, k = 1,\dots,W, l = 1,\dots H, n = 1,\dots,N, t = 1,\dots,T \tag{7.9}$$

$$y_{ijnt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 1,\dots H, n = 1,\dots,N, t = 1,\dots,T \tag{7.10}$$

$$b_{ijnt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 1,\dots H, n = 1,\dots,N, t = 2,\dots,T \tag{7.11}$$

The objective function minimizes the total number of relocations. Constraint (7.1) imposes that each container is either within the bay or retrieved. Constraint (7.2) makes sure that each position (i, j) may be occupied by at most one container. Constraint (7.3) prevents gaps within stacks. This makes sure that containers at positions (i, j') with $j' > j$ are relocated if a container is retrieved from position (i, j) and that no containers are relocated to suspended positions. Constraint (7.4) ensures the consistency of the bay over time. It links the layout at period t with the layout at period $t + 1$ via the executed retrieval and relocations. Constraint (7.5) makes sure that container $n = t$ is retrieved from the bay in period t.

Constraint (7.6) is supposed to impose the LIFO order among two or more relocation containers. It makes sure that container n cannot be stacked below container n' after the relocation, if container n was stacked below container n' before the relocation. But, this constraint is over-restrictive and makes it impossible to relocate two containers to the same stack. Consider the case where no container is relocated from position (i, j) to position (k, l) in period t. Then, $\sum_{n=1}^{N} x_{i,j,k,l,n,t}$ on the left hand side of the constraint equals 0. Constraint (7.12) displays the resulting constraint which imposes that at most one container may be relocated from a position above (i, j) to a position above (k, l). But, it has to be possible to relocate two containers to the same stack if they respect the LIFO order.

$$1 \geq \sum_{n=1}^{N} \sum_{j'=j+1}^{H} \sum_{l'=l+1}^{H} x_{ij'kl'nt} \qquad \forall i, j, k, l, t | \sum_{n=1}^{N} x_{ijklnt} = 0 \tag{7.12}$$

Constraint (7.7) limits relocations to containers stored in the same stack as the retrieval container. However, it does not impose that only containers above the target container are relocated (nor does any other constraint). Consequently, assumption A8 limiting relocations to blocking container is not represented in their model. Constraint (7.8) reduces the number of variables by excluding variables that relocate containers into the stack from which they have been retrieved. Constraints (7.9) to (7.10) define the binary variables.

7.2 Model improvements

This section presents our reformulation of their model. It corrects the errors of the previous model, removes superfluous variables, presents a preprocessing step to fix several variables and cuts based on upper bounds on the number of relocations.

7.2.1 Reformulation

We formulate constraints to represent the LIFO order among several retrieval containers and assumption A8 correctly. We replace Constraint (7.6) by Constraint (7.13) which allows M relocations if $\sum_{n=1}^{N} x_{ijklnt} = 0$ and imposes the LIFO constraint otherwise. We set M to $H - 1$ since the maximum number of relocations per period is limited by the bay height and equals $H - 1$.

$$M \cdot \left(1 - \sum_{n=1}^{N} x_{ijklnt}\right) \geq \sum_{n=1}^{N} \sum_{j'=j+1}^{H} \sum_{l'=l+1}^{H} x_{ij'kl'nt} \qquad \forall i, j, k, l, t \tag{7.13}$$

We replace Constraint (7.7) by Constraints (7.14) and (7.15). Constraint (7.14) forbids all relocations out of stacks other than the stack containing the target container; Constraint (7.15) forbids all relocations out of positions below the target container. Adding M with $M = H - 1$ makes sure that these constraints are not over-restrictive.

$$M \cdot \sum_{j=1}^{H} y_{ijtt} \geq \sum_{j=1}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{n=t+1}^{N} x_{ijklnt} \qquad \forall i, t \tag{7.14}$$

$$M \cdot y_{ijtt} + \sum_{j'=1}^{j} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{n=t+1}^{N} x_{ij'klnt} \leq M \qquad \forall i, j, t \tag{7.15}$$

The model may be improved further by removing unnecessary variables and constraints. We know that at each period $t = 1, \ldots, T$ container $n = t$ has to be retrieved. This implies that at the beginning of period t, only containers $n = t, \ldots, T$ are in the bay and that only containers $n = t + 1, \ldots, T$ may be relocated. We define variables y_{ijnt} only for $n = t$, variables b_{ijnt} only for $n \geq t$ and variables x_{ijklnt} only for $n > t$. In addition, period T may be excluded from the model since the only remaining container has to be retrieved and no relocations may occur. We use variables y_{ijnt} to impose that container $n = t$ is retrieved at period t and get rid of parameters v_{nt} and Constraints (7.1) and (7.5) used for the same purpose. Containers may only be relocated from above the target container. Hence, no relocations from tier 1 may occur and we dismiss variables x_{i1klnt} with $j = 1$. The modified model is presented below.

$$\sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t=1}^{T-1} \sum_{n=t+1}^{N} x_{ijklnt}$$

s.t.

$$\sum_{n=t}^{N} b_{ijnt} \leq 1 \qquad \forall i = 1, \ldots, W, j = 1, \ldots H, t = 1, \ldots, T-1 \tag{7.16}$$

$$\sum_{n=t}^{N} b_{ijnt} \geq \sum_{n=t}^{N} b_{ij+1nt} \qquad \forall i = 1, \ldots, W, j = 1, \ldots H-1, t = 1, \ldots, T \tag{7.17}$$

$$b_{ijnt+1} = b_{ijnt} + \sum_{k=1}^{W} \sum_{l=2}^{H} x_{klijnt} - \sum_{k=1}^{W} \sum_{l=1}^{H} x_{ijklnt} \qquad \forall i = 1, \ldots, W, j = 1, \ldots H, t = 1, \ldots, T-1, n = t+1, \ldots, N \tag{7.18}$$

$$b_{ijnt} - y_{ijtt} = 0 \qquad \forall i = 1, \ldots, W, j = 1, \ldots H, t = 1, \ldots, T-1, n = t \tag{7.19}$$

$$\sum_{i=1}^{W}\sum_{j=1}^{H} y_{ijtt} = 1 \quad \forall t = 1,\dots,T-1 \tag{7.20}$$

$$M \cdot \left(1 - \sum_{n=t+1}^{N} x_{ijklnt}\right) \geq \sum_{n=t+1}^{N}\sum_{j'=j+1}^{H}\sum_{l'=l+1}^{H} x_{ij'kl'nt} \quad \forall i = 1,\dots,W, j = 2,\dots H-1, k = 1,\dots,W, l = 1,\dots H-1, t = 1,\dots,T-1 \tag{7.21}$$

$$M \cdot \sum_{j=1}^{H} y_{ijtt} \geq \sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt} \quad \forall i = 1,\dots,W, t = 1,\dots,T-1 \tag{7.22}$$

$$M \cdot y_{ijtt} + \sum_{j'=2}^{j}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ij'klnt} \leq M \quad \forall i = 1,\dots,W, j = 3,\dots H, t = 1,\dots,T-1 \tag{7.23}$$

$$x_{ijilnt} = 0 \quad \forall i = 1,\dots,W, j = 2,\dots H, l = 1,\dots H, t = 1,\dots,T-1, n = t+1,\dots,N \tag{7.24}$$

$$x_{ijklnt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 2,\dots H, k = 1,\dots,W, l = 1,\dots H, t = 1,\dots,T-1, n = t+1,\dots,N \tag{7.25}$$

$$y_{ijtt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 1,\dots H, t = 1,\dots,T-1 \tag{7.26}$$

$$b_{ijnt} \in \{0,1\} \quad \forall i = 1,\dots,W, j = 1,\dots H, t = 1,\dots,T, n = t,\dots,N \tag{7.27}$$

The objective function minimizes the total number of relocations. Constraints (7.16) and (7.17) adapt Constraints (7.2) and (7.3) to the reduced number of variables. They make sure that every position is occupied by at most one container and prevent gaps within stacks. Constraint (7.4) is split into Constraints (7.18) and (7.19) since variables y_{ijnt} and x_{ijklnt} are defined for different time periods. They link the bay layout from one period to the next. Constraint (7.21) imposes the LIFO order. Constraints (7.22) and (7.23) limit relocations to containers placed above the target container[1]. Constraint (7.24) is unchanged and forbids relocations to the stack from which containers are retrieved. Constraints (7.25) to (7.27) define the reduced variable domains of binary variables.

Table 7.1 roughly compares the size of the initial and the reformulated model. It indicates the number of variables and constraints depending on the bay width W, the bay height H

[1]A tighter possiblity to limit relocations to blocking containers is $\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt} \leq \sum_{j'=1}^{j-1} y_{ij'tt}$ $\forall i = 1,\dots,W, j = 2,\dots H, t = 1,\dots,T-1$. The results presented in the sequel, however, are for the model with Constraints (7.22) and (7.23)

Table 7.1: Size of the initial and the reformulated model

BIP from Caserta et al. (2012)							
Inst *S-W*	H	W	N	y_{ijnt} $W \cdot H \cdot N^2$	b_{ijnt} $W \cdot H \cdot N^2$	x_{ijklnt} $W^2 \cdot H^2 \cdot N^2$	Constraints[*1]
3-3	5	3	9	1 215	1 215	18 225	3 699
3-8	10	8	24	46 080	46 080	3 686 400	204 864
4-4	6	4	16	6 144	6 144	147 456	16 704
4-7	9	7	28	49 392	49 392	3 111 696	165 816
5-4	6	4	20	9 600	9 600	230 400	22 960
5-10	12	10	50	300 000	300 000	36 000 000	1 037 500
6-6	8	6	36	62 208	62 208	2 985 984	151 416
10-10	12	10	100	1 200 000	1 200 000	144 000 000	2 685 000

[*1]Constraints: $2 \cdot N^2 + W \cdot N + 2 \cdot W \cdot H \cdot N + W \cdot H \cdot N^2 + W^2 \cdot H^2 \cdot N$

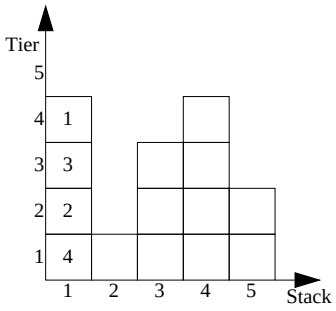
Reformulated BIP							
Inst *S-W*	H	W	N	y_{ijnt} $W \cdot H \cdot N$	b_{ijnt} $W \cdot H \cdot N^2/2$	x_{ijklnt} $W^2 \cdot H^2 \cdot N^2/2$	Constraints[*2]
3-3	5	3	9	135	608	9113	3209
3-8	10	8	24	1 920	23 040	1 843 200	184 536
4-4	6	4	16	384	3 072	73 728	13 904
4-7	9	7	28	1 764	24 696	1 555 848	143 108
5-4	6	4	20	480	4 800	115 200	18 340
5-10	12	10	50	6 000	150 000	18 000 000	894 550
6-6	8	6	36	1 728	31 104	1 492 992	121 212
10-10	12	10	100	12 000	600 000	72 000 000	2 089 100

[*2]Constraints: $N + W \cdot N + 4 \cdot W \cdot H \cdot N + W \cdot H \cdot N^2/2 + W^2 \cdot H^2 \cdot N$

and the number of containers N. Remember that $T = N$ and $H = S + 2$. It illustrates the resulting problem size for different instances. For both models, the number of variables increases faster than the number of constraints. The initial model uses approximately T times as many variables y_{ijnt} and approximately twice as many variables b_{ijnt} and x_{ijklnt} than the reformulated model. It needs approximately $T(2T-1) + W \cdot H \cdot T(1/2T-2)$ more constraints.

7.2.2 Preprocessing

We use information provided by the initial layout to fix several variables x_{ijklnt} and b_{ijnt} to zero or one. We use (i_n, j_n) to refer to the initial position of container n. Let π_n be the period when container n is removed from its initial position for the first time. Since only blocking containers may be relocated, this happens either when the container is retrieved or when it is relocated to free a target container beneath. Let $\pi_{(i,j)}$ be the period when position (i, j) is empty for the first time. Either the positon is empty in the initial layout or when the container initially located there is removed. Equations (7.28) and (7.29) define parameters π_n



Stack 1			
$\pi_{(1,1)}$	5	π_1	1
$\pi_{(1,2)}$	3	π_2	2
$\pi_{(1,3)}$	3	π_3	2
$\pi_{(1,4)}$	2	π_4	4
$\pi_{(1,5)}$	1		

Figure 7.1: Preprocessing: computation of π_n and $\pi_{(i,j)}$

and $\pi_{(i,j)}$ accordingly. Figure 7.1 illustrates the computation of π_n and $\pi_{(i,j)}$ on an example. It displays the initial positions of containers 1 to 4 in stack 1 and the resulting values for π_1 to π_4 and $\pi_{(1,1)}$ to $\pi_{(1,5)}$.

$$\pi_n = \min n' \mid n' \text{ is placed at any position } (i_n, 1) \text{ to } (i_n, j_n) \tag{7.28}$$

$$\pi_{(i,j)} = \begin{cases} \pi_n + 1 & \text{if container } n \text{ is placed at position } (i,j), \\ 1 & \text{if position } (i,j) \text{ is empty} \end{cases} \tag{7.29}$$

With the knowledge of the initial position (i_n, j_n) for each container n and the information provided by π_n and $\pi_{(i_n,j_n)}$ we may fix the following variables:

- Container n is not relocated prior to period π_n and only from position (i_n, j_n) in period π_n:
  $x_{ijklnt} = 0 \qquad \forall n, i, j, k, l, t < \pi_n$
  $x_{ijklnt} = 0 \qquad \forall n, i \neq i_n, j \neq j_n, k, l, t = \pi_n$
- Container n is at position (i_n, j_n) and nowhere else until period π_n:
  $b_{i_n j_n nt} = 1 \qquad \forall n, i_n, j_n, t \leq \pi_n$
  $b_{ijnt} = 0 \qquad \forall n, i \neq i_n, j \neq j_n, t \leq \pi_n$
- No container n' may be relocated to position (i_n, j_n) prior to period $\pi_{(i_n,j_n)}$ since it already contains container n:
  $x_{iji_n j_n n't} = 0 \qquad \forall n, n', i, j, i_n, j_n, t < \pi_{(i_n,j_n)}$
- If $\pi_n = n$, container n is never relocated and retrieved from its initial position (i_n, j_n). In this case, only containers in stack i_n and above j_n may be relocated in period π_n:
  $x_{ijklnt} = 0 \qquad \forall n, i \neq i_n, j, t = \pi_n | n = \pi_n$
  $x_{i_n jklnt} = 0 \qquad \forall n, i_n, j \leq j_n, t = \pi_n | n = \pi_n$
- If $\pi_n = n$, we know that position (i_n, j_n) and all positions above have to be empty in period $n + 1$:
  $b_{i_n,jnt} = 0 \qquad \forall n, i_n, j \geq j_n, t = \pi_n + 1 | n = \pi_n$

We may fix further variables by using the fact that at each period exactly one container is retrieved. Consequently, the number of containers remaining in the bay at the beginning of period t is given by $N_t = N + 1 - t$. The number of containers in the bay limits the maximum stack height:

– At period t, tiers at height $h > N_t$ may not be occupied:
$b_{ijnt} = 0 \qquad \forall t, i, j > N_t$

– At period t, relocation containers can only be retrieved from tiers $h \leq N_t$:
$x_{ijklnt} = 0 \qquad \forall t, i, j > N_t, k, l, n$

– At period t, relocation containers can only be put into tiers $h \leq N_{(t+1)} = N_t - 1$:
$x_{ijklnt} = 0 \qquad \forall t, i, j, k, l > N_t - 1, n$

7.2.3 Cuts

We use the upper bounds presented in Section 6.3 to add cuts to the binary programming model. UB represents the solution of Heuristic HC. $R^t_{\max,1}$ the maximum number of relocations for period t to be able to obtain a solution better than UB.

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt} \leq R^t_{\max,1} \qquad \forall t = 1, \ldots, T-1 \tag{7.30}$$

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{t=1}^{T-1}\sum_{n=t+1}^{N} x_{ijklnt} \leq UB - 1 \tag{7.31}$$

7.3 Model comparison

Unfortunately, we cannot compare our binary programming model with other models from literature. They either represent a slightly different problem (Caserta et al. (2012), Petering and Hussein (2013)) or run experiments on different instances (Tang et al. (2012)). Instead, this section evaluates the benefits of the preprocessing step and of cuts. It compares the optimal solution of different models as well as the lower bound of their linear relaxations. Linear relaxations are also compared against the lower bound LB obtained from the initial bay layout. We compare the following models:

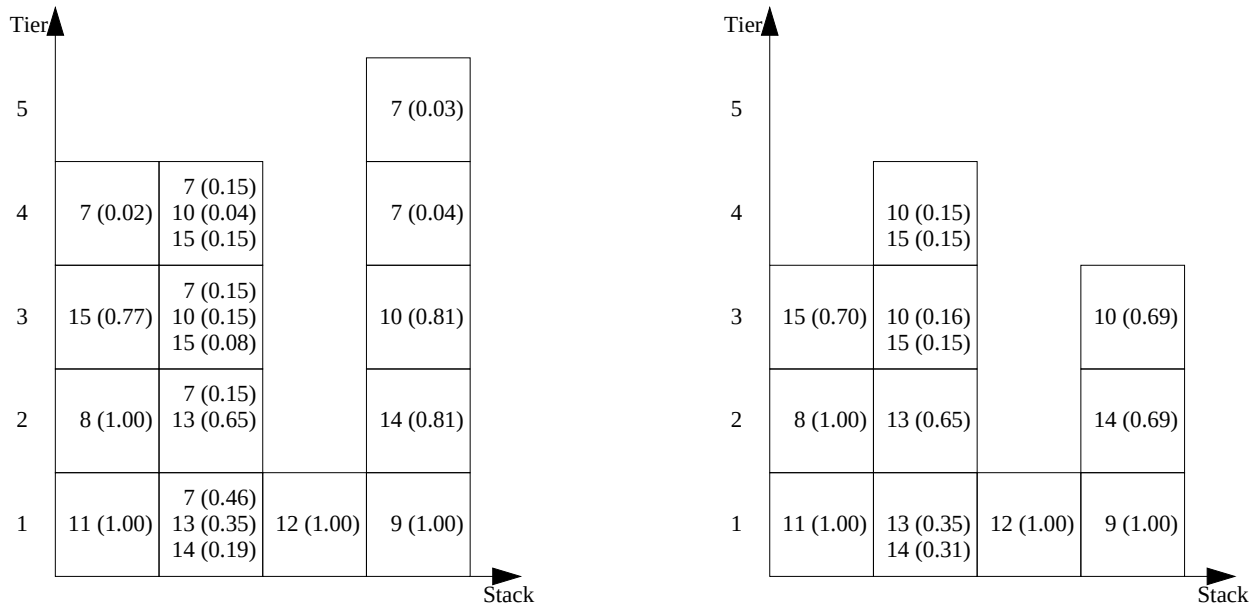
– **Reformulated model (BIP1):** model presented in Section 7.2.1;

– **Reformulated model with preprocessing (BIP2):** identical to BIP1 with preprocessing to fix variables x_{ijklnt} and b_{ijnt} if possible;

– **Reformulated model with preprocessing and cuts (BIP3):** identical to BIP2 with additional cuts limiting the number of relocations.

Experiments are carried out on a computer with Intel(R) Xeon(R) CPU clocked at 2.67GHz (dual core), 3.48GB RAM and operating with Windows XP Professional. We use Cplex 12.1 to solve the binary programming models and their linear relaxations. Experiments are run for the instances presented in section 6.4. We limit the run time to 60 minutes per instance.

Table 7.2 shows the results for BIP1 to BIP3 in an aggregate way for instance sets 3-3 to 5-6. It indicates the number of trivial and non-trivial instances per set. For all three models, it displays the number of trivial and non-trivial instances that were solved to optimum and



(a) Fractional layout at period 7

(b) Two fractional layout at period 8

Figure 7.2: Two fractional solutions indicating container locations and values of variables $b_{ijnt} > 0$ (in brackets)

average solution times for solved instances. It also displays the number of instances that could not be solved because of time or memory limits. Results for instances 5-7 to 10-10 are not displayed since all models run out of memory.

Results show that BIP2 and BIP3 outperform BIP1 by far. Solution times are decreased by 8% for smaller instances to 99% for bigger instances. BIP2 and BIP3 also solve more instances of sets 4-5 to 5-4 and bigger instances of sets 5-5 and 5-6. The reason is that fewer instances are aborted because of time or memory limits. BIP2 and BIP3 perform similarly with regard to the number of solved instances and solution times. These results show that the preprocessing step fixing several variables is very effective. The added cuts have no impact, since Constraint (7.22) already limits the number of relocations per period to M. For all sets (except set 5-5 with one trivial instance), trivial instances are solved faster than non-trivial instances. But, they are not solved immediately. It is hence beneficial to check if the heuristic solution is optimal before starting the solution process. This check can be done with the optimality criterion presented in Section 6.3.

When relaxing binary conditions, one container may be located in several positions and several containers may be placed into the same position. In addition, the LIFO order does not have to be respected, all containers and not only blocking ones may be relocated and not all blocking containers have to be relocated. Only constraints preventing gaps within stacks and constraints imposing the consistency of the bay over time hold. Figure 7.2 illustrates feasible relocations in the linear relaxed case. It presents fractional layouts at period 7 and at period 8 after retrieving container 7. They are determined by the values of variables b_{ijnt}. Instead of relocating containers 10, 13 and 15 from above container 7 in stack 3 to another stack, containers 10 and 14 (from stack 5) and container 15 (from stack 1) are relocated into stack 3 to satisfy $\sum_n b_{ijnt} \geq \sum_n b_{ij+1nt}$.

Table 7.2: Performance comparison of BIP1, BIP2 and BIP3 for trivial (tr) and non-trivial (ntr) instances

	Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4	4-5	4-6	4-7	5-4	5-5	5-6	Total
	Nb. trivial	27	26	28	36	33	32	8	11	11	9	4	1	2	228
	Nb. non-trivial	13	14	12	4	7	8	32	29	29	31	36	39	38	292
BIP 1	Nb. solved tr	27	26	28	36	33	32	8	11	11	9	4	0	0	225
	Avg. time tr [s]	0.2	0.6	1.8	5.2	12.0	29.3	6.6	156.5	615.0	615.6	840.8	n.a.	n.a.	207.6
	Nb. solved ntr	13	14	12	4	7	8	32	28	19	9	27	4	0	177
	Avg. time ntr [s]	0.2	0.8	2.3	8.6	20.8	93.4	12.3	457.5	479.8	894.0	914.7	2080.2	n.a.	413.7
	Nb. time ntr	0	0	0	0	0	0	0	1	6	22	9	13	0	51
	Nb. mem ntr	0	0	0	0	0	0	0	0	4	0	0	23	40	67
BIP 2	Nb. solved tr	27	26	28	36	33	32	8	11	11	9	4	1	2	228
	Avg. time tr [s]	0.1	0.3	0.8	2.1	3.7	7.0	1.1	3.4	7.9	15.3	20.1	922.6	36.6	78.6
	Nb. solved ntr	13	14	12	4	7	8	32	29	27	24	36	30	12	248
	Avg. time ntr [s]	0.1	0.4	1.2	4.5	6.6	11.2	2.0	15.4	24.6	84.4	329.3	356.5	828.9	128.1
	Nb. time ntr	0	0	0	0	0	0	0	0	2	1	0	9	22	34
	Nb. mem ntr	0	0	0	0	0	0	0	0	0	6	0	0	4	10
BIP 3	Nb. solved tr	27	26	28	36	33	32	8	11	11	9	4	1	2	228
	Avg. time tr [s]	0.2	0.5	1.3	3.0	5.2	10.2	1.5	5.2	11.0	20.8	15.1	148.8	57.4	21.6
	Nb. solved ntr	13	14	12	4	7	8	32	29	28	24	34	30	10	245
	Avg. time ntr [s]	0.2	0.5	1.3	4.9	6.8	24.1	2.1	27.2	32.2	66.4	209.9	296.1	726.1	107.5
	Nb. time ntr	0	0	0	0	0	0	0	0	1	2	2	9	23	37
	Nb. mem ntr	0	0	0	0	0	0	0	0	0	5	0	0	5	10

Table 7.3 compares the lower bounds obtained from the linear relaxations LR_BIP1 to LR_BIP3 of the binary models BIP1 to BIP3 with the lower bound LB obtained from the initial layout. We use the rounded up solution of linear relaxations as their lower bounds. For each instance set, the table indicates the average lower bound and the average and maximum gap to the optimal or best known solution.

Table 7.3: Comparison of lower bounds obtained from the initial layout LB and by linear relaxations of binary models LR_BIP1 to LR_BIP3

	Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4
LB	Avg. rel	4.7	5.9	6.8	8.3	9.1	10.5	9.4
	Avg. gap [%]	-4	-4	-3	-1	-1	-2	-7
	Max gap [%]	-17	-29	-14	-18	-15	-14	-30
LR_BIP 1	Avg. rel	4.6	5.7	6.6	8.0	8.8	9.9	9.1
	Avg. gap [%]	-5	-4	-4	-2	-2	-4	-8
	Max gap [%]	-20	-29	-17	-18	-20	-14	-25
LR_BIP 2	Avg. rel	4.8	5.9	6.8	8.3	9.1	10.5	9.6
	Avg. gap [%]	-2	-2	-2	-1	-1	-2	-5
	Max gap [%]	-17	-29	-14	-18	-9	-14	-23
LR_BIP 3	Avg. rel	4.8	5.9	6.8	8.3	9.1	10.5	9.6
	Avg. gap [%]	-2	-2	-2	-1	-1	-2	-5
	Max gap [%]	-17	-29	-14	-18	-9	-14	-23

	Inst. set	4-5	4-6	4-7	5-4	5-5	5-6	Total
LB	Avg. rel	12.2	13.2	15.2	13.6	17.0	20.1	11.2
	Avg. gap [%]	-6	-6	-5	-12	-10	-9	-5
	Max gap [%]	-21	-19	-15	-24	-25	-20	-30
LR_BIP 1	Avg. rel	11.3	12.2	14.1	12.5	15.8	18.4	10.5
	Avg. gap [%]	-10	-10	-10	-16	-14	-15	-8
	Max gap [%]	-36	-33	-33	-39	-41	-33	-41
LR_BIP 2	Avg. rel	12.2	13.3	15.3	13.9	17.2	20.2	11.3
	Avg. gap [%]	-5	-4	-4	-9	-8	-8	-4
	Max gap [%]	-18	-14	-15	-22	-25	-18	-29
LR_BIP 3	Avg. rel	12.2	13.3	15.3	13.9	17.2	20.2	11.3
	Avg. gap [%]	-5	-4	-4	-9	-8	-8	-4
	Max gap [%]	-18	-14	-15	-22	-25	-18	-29

LR_BIP1 provides a very weak lower bound since most constraints are relaxed for linear variables. It has the highest average and maximum gap for almost all instances. It is outperformed by LB on 128 instances, obtains the same bound on 377 instances and obtains a better bound on only 15 instances.

BIP2 fixes variables b_{ijnt} and x_{ijklnt} to 0 or 1 for each container until it is retrieved or relocated for the first time. This limits the solution space since several variables are forced

to take integer values and as long as variables take integer values, constraints are tight. LR_BIP2 performs similar to LB. However, LB does not consider containers after their retrieval / first relocation whereas LR_BIP2 considers them in a relaxed way. It provides a tighter bound than LB on 90 instances and the same bound on 430 instances. LR_BIP3 is identical to LR_BIP2 for all instances since cuts determined for the binary case are too weak for the relaxed case.

7.4 Conclusion

This chapter improved an existing binary model for the container relocation problem where only blocking containers are relocated. We reduced the model size by removing superfluous variables and corrected two errors in the existing formulation. We also introduced a preprocessing step and cuts. The preprocessing step uses information provided by the initial layout to fix several variables. Cuts use the heuristic solution to limit the number of relocations.

Experimental results showed that the preprocessing step improves the performance of the model: run times decrease considerably, more instances can be solved and the lower bound obtained by the linear relaxation is tightened. Cuts are too weak and do not improve the performance of the model.

It would be interesting to compare the performance of the reformulated model with other models from literature. Unfortunately, this was not possible since they either represent a slightly different problem (Caserta et al. (2012), Petering and Hussein (2013)) or run experiments on different instances (Tang et al. (2012)).

The reformulated binary model with the preprocessing step solves small and medium instances in reasonable time. Bigger instances however cannot be solved due to time and memory. Alternative solution approaches have to be designed for bigger instances. The next two chapters present an exact and a heuristic branch and price approach.

Chapter 8

Branch and price approach for CRP

The binary programming models presented in Chapter 7 are impractical for bigger instances. They require a huge number of variables and provide weak linear relaxations. Column generation is suitable to solve linear programming models with a huge number of variables and to determine tight linear relaxations of integer programming models.

This chapter presents a branch and price approach for the container relocation problem where only blocking containers are relocated. Section 8.1 presents our column generation approach. It details the decomposition of the binary model into a master problem and a subproblem. It describes a column generation approach where both the master problem and the subproblem are solved with an IP solver. It also interprets the information provided by dual variables and shows how it can be used for a new upper bound on the number of relocations. Section 8.2 presents an enumerative subproblem and two mechanisms used to reduce the number of columns to enumerate. Section 8.3 details the branching procedure chosen for branch and price. Section 8.4 compares our different column generation approaches and evaluates the branch and price. Results are also compared to the results obtained with the binary model of the previous chapter. Section 8.5 concludes the chapter.

8.1 Column generation for CRP

Algorithm 8.1 recalls the general procedure of column generation. A restricted linear programming master problem (a linear reformulation of the initial problem) is initialized with all constraints and a subset of variables called columns. After solving the restricted master problem, a pricing subproblem determines columns that could improve the objective value of the restricted master problem (columns with negative reduced costs). Columns are determined based on values of dual variables of the restricted master problem. These

Algorithm 8.1 General column generation procedure

initialize restricted master problem (RMP)
repeat
  add column(s) with negative reduced costs to RMP
  solve RMP
  execute subproblem for updated values of dual variables
until no column with negative reduced costs found
return optimal solution for RMP

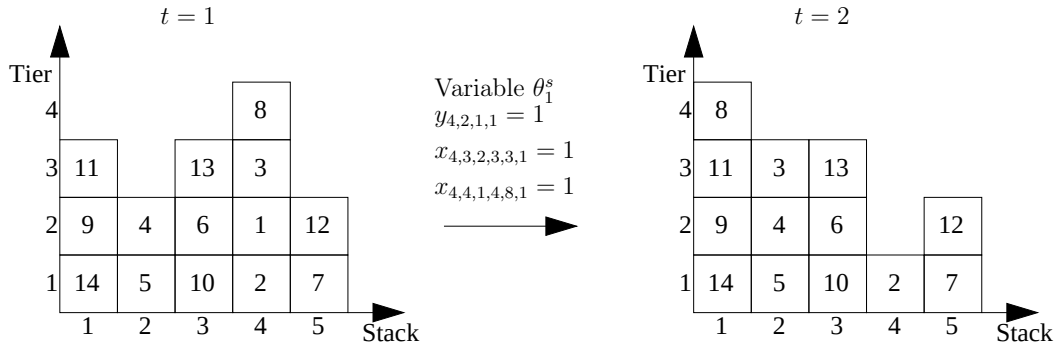


Figure 8.1: The way column θ_1^s transforms the bay (if applied)

columns are then added to the restricted master problem. The updated model is solved and the subproblem is executed for the updated values of dual variables. This iterative process stops if no more columns with negative reduced costs exist. In this case, an optimal linear solution is obtained which is a linear relaxation of the initial problem.

We keep physical constraints on the bay (e.g., no holes, consistency over time) in the master problem and use the subproblem to generate a series of movements with one retrieval and several relocations. We use columns θ_t^s to represent the retrieval of container $n = t$ and relocations of any containers stacked above. Binary parameters y_{ijnt}^s and x_{ijklnt}^s indicate which retrieval and which relocations have to be executed for column θ_t^s. S_t is the number of columns θ_t^s that retrieve container t in period t and

$$\theta_t^s = \begin{cases} 1 & \text{if the column is applied,} \\ 0 & \text{otherwise.} \end{cases} \tag{8.1}$$

Figure 8.1 illustrates how columns θ_t^s represent retrievals and relocations and how they transform the bay if applied. If θ_1^s is applied in period 1, it retrieves container 1 from position $(4, 2)$, relocates container 3 from position $(4, 3)$ to $(2, 3)$ and container 8 from position $(4, 4)$ to $(1, 4)$. This leads to the bay depicted for period 2.

8.1.1 Master problem

The master problem is a reformulation of the binary model presented in Section 7.2.1.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t=1}^{T-1} \sum_{n=t+1}^{N} \sum_{s=1}^{S_t} x_{ijklnt}^s \cdot \theta_t^s$$

s.t.

$$\sum_{n=t}^{N} b_{ijnt} \leq 1 \qquad \forall i = 1, \ldots, W, j = 1, \ldots, H, t = 1, \ldots, T-1 \tag{8.2}$$

$$\sum_{n=t}^{N} b_{ijnt} \geq \sum_{n=t}^{N} b_{ij+1nt} \quad \forall i = 1,\dots,W, j = 1,\dots,H-1, t = 1,\dots,T \tag{8.3}$$

$$b_{ijnt+1} = b_{ijnt} + \sum_{s=1}^{S_t}\sum_{k=1}^{W}\sum_{l=2}^{H} \theta_t^s \cdot x_{klijnt}^s - \sum_{s=1}^{S_t}\sum_{k=1}^{W}\sum_{l=1}^{H} \theta_t^s \cdot x_{ijklnt}^s \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1, n = t+1,\dots,N \tag{8.4}$$

$$b_{ijtt} - \sum_{s=1}^{S_t} \theta_t^s \cdot y_{ijtt}^s = 0 \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1 \tag{8.5}$$

$$\sum_{s=1}^{S_t} \theta_t^s = 1 \quad \forall t = 1,\dots,T-1 \tag{8.6}$$

$$\theta_t^s \geq 0 \quad \forall t = 1,\dots,T-1, s = 1,\dots,S_t \tag{8.7}$$

$$b_{ijnt} \geq 0 \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T, n = t,\dots,N \tag{8.8}$$

The objective function minimizes the total number of relocations. Constraint (8.2) makes sure that each position (i, j) is occupied by at most one container. Constraint (8.3) prevents gaps within stacks. Constraint (8.4) links the bay layout at period t with the layout at period $t + 1$ via the executed relocations. Constraint (8.5) makes sure that container t is retrieved from the bay in period t. Constraint (8.6) imposes that one column is chosen per period. Constraints (8.7) and (8.8) represent the domain definitions of the linear variables. All b_{ijn1} with $t = 1$ are parameters which indicate the initial configuration of the bay. We also fix variables b_{ijnt} with the preprocessing step presented in Section 7.2.1.

Constraint (8.9) does not result from the reformulation. It is a cut to enhance the quality of the model. It imposes that position (i, j) has to be empty at period $t + 1$ if container t is retrieved from a position (i, j') with $j' \leq j$. Because of the acceptable number of cuts, all cuts are added to the model from the beginning on.

$$\sum_{j'=1}^{j} b_{ij'tt} + \sum_{n=t+1}^{N} b_{ijnt+1} \leq 1 \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1 \tag{8.9}$$

8.1.2 Pricing subproblem

The pricing subproblem generates new columns to be added to the master problem. These columns have to be feasible with regard to the problem description. In addition, they must have the potential to reduce the value of the objective function of the restricted master problem. In other words, they must have negative reduced costs.

To obtain the reduced cost vector of a column, we formulate the dual of the restricted master problem. Dual variables of the restricted master problem are: α_{ijt} for Constraint (8.2), β_{ijt} for Constraint (8.3), γ_{ijnt} for Constraint (8.4), δ_{ijt} for Constraint (8.5) and μ_t for Constraint (8.6).

$$\max \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{t=1}^{T-1}\alpha_{ijt} + \sum_{t=1}^{T-1}\mu_t$$

s.t.

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt}\cdot(-\gamma_{ijnt}+\gamma_{klnt}) - \sum_{i=1}^{W}\sum_{j=1}^{H} y_{ijtt}\cdot\delta_{ijt} + \mu_t \leq \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt} \quad \forall t = 1,\dots,T-1 \tag{8.10}$$

$$\alpha_{ijt} + \beta_{ijt} - \beta_{ij-1t} + \gamma_{ijnt} - \gamma_{ijnt-1} + \delta ijt \leq 0 \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1, n = t,\dots,N \tag{8.11}$$

$$\alpha_{ijt} \leq 0 \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1 \tag{8.12}$$

$$\beta_{ijt} \geq 0 \quad \forall i = 1,\dots,W, j = 1,\dots,H-1, t = 1,\dots,T-1 \tag{8.13}$$

$$\gamma_{ijnt} \in \mathbb{R} \quad \forall i = 1,\dots,W, j = 2,\dots,H, t = 1,\dots,T-2, n = t+1,\dots,N \tag{8.14}$$

$$\delta_{ijt} \in \mathbb{R} \quad \forall i = 1,\dots,W, j = 1,\dots,H, t = 1,\dots,T-1 \tag{8.15}$$

$$\mu_t \in \mathbb{R} \quad \forall t = 1,\dots,T-1 \tag{8.16}$$

The reduced cost vector of a column for period t is given by Equation (8.17).

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt}\cdot(1+\gamma_{ijnt}-\gamma_{klnt}) + \sum_{i=1}^{W}\sum_{j=1}^{H} y^s_{ijtt}\cdot\delta_{ijt} - \mu_t \tag{8.17}$$

The subproblem is formulated for each period t. It determines the column with the smallest negative reduced costs for period t respecting all constraints from the initial problem description.

$$\min \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N}(1+\gamma_{ijnt}-\gamma_{klnt})\cdot x_{ijklnt}+\sum_{i=1}^{W}\sum_{j=1}^{H}\delta_{ijt}\cdot y_{ijtt}-\mu_t$$

s.t.

$$\sum_{i=1}^{W}\sum_{j=1}^{H} y_{ijtt}=1 \tag{8.18}$$

$$M\cdot\left(1-\sum_{n=t+1}^{N} x_{ijklnt}\right)\geq\sum_{n=t+1}^{N}\sum_{j'=j+1}^{H}\sum_{l'=l+1}^{H} x_{ij'kl'nt} \tag{8.19}$$
$$\forall i=1,\ldots,W, j=2,\ldots H-1, k=1,\ldots,W, l=1,\ldots H-1$$

$$M\cdot\sum_{j=1}^{H} y_{ijtt}\geq\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt} \tag{8.20}$$
$$\forall i=1,\ldots,W$$

$$M\cdot y_{ijtt}+\sum_{j'=2}^{j}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ij'klnt}\leq M \tag{8.21}$$
$$\forall i=1,\ldots,W, j=3,\ldots H$$

$$x_{ijilnt}=0 \tag{8.22}$$
$$\forall i=1,\ldots,W, j=2,\ldots H, l=1,\ldots H, n=t+1,\ldots,N$$

$$y_{ijtt}+\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt}\leq 1 \tag{8.23}$$
$$\forall i=1,\ldots,W, j=2,\ldots,H$$

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{n=t+1}^{N} x_{ijklnt}\leq 1 \tag{8.24}$$
$$\forall k=1,\ldots,W, l=1,\ldots,H$$

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H} x_{ijklnt}\leq 1 \tag{8.25}$$
$$\forall n=t+1,\ldots,N$$

$$1-y_{ijtt}-\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{T} x_{ijklnt}+\sum_{h=j+1}^{j'-1}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{T} x_{ihklnt}\geq\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{T} x_{ij'klnt}$$
$$\forall i=1,\ldots,W, j=2,\ldots,H-2, j'=j+2,\ldots,H \tag{8.26}$$

$$1 - \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{n=t+1}^{T} x_{ijklnt} + \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{h=j+1}^{j'-1}\sum_{n=t+1}^{T} x_{ijkhnt} \geq \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{n=t+1}^{T} x_{ijkl'nt} \quad (8.27)$$
$$\forall k = 1, \ldots, W, l = 1, \ldots, H-2, l' = l+2, \ldots, H$$

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x_{ijklnt} \leq R^t_{\max,1} \quad (8.28)$$

$$y_{ijtt} \in \{0,1\} \quad \forall i = 1, \ldots, W, j = 1, \ldots, H \quad (8.29)$$

$$x_{ijklnt} \in \{0,1\} \quad \forall i = 1, \ldots, W, j = 2, \ldots, H, k = 1, \ldots, W, l = 1, \ldots, H, n = t+1, \ldots, N \quad (8.30)$$

The objective function minimizes the reduced cost. If the objective value is not negative no columns with negative reduced costs exist for period t. Constraints (8.18) to (8.22) are taken from the initial binary formulation. Constraint (8.18) makes sure that container $n = t$ is retrieved at period t. Constraint (8.19) imposes the LIFO order between containers being relocated into the same stack. Constraints (8.20) and (8.21) ensure that only blocking containers are relocated. Constraint (8.22) prevents relocating a container into the stack from which it has been retrieved.

Constraints (8.23) to (8.29) are added to represent physical constraints to obtain a column that is feasible with regard to the initial problem description. Constraint (8.23) makes sure that at most one container (retrieval or relocation) is taken out of each position. Constraint (8.24) ensures that at most one relocation container is put into each position. Constraint (8.25) imposes that each container is relocated at most once within period t. Constraint (8.26) imposes that retrieval and relocation containers are taken from adjacent tiers. If container n is retrieved from position (i, j) and container n' from position (i, j'), we obtain $y_{ijtt} = 1$ or $x_{ijklnt} = 1$ and $x_{i'jkln't} = 1$. To satisfy the inequality at least one container has to be retrieved from each position (i, h) with $h = j+1, \ldots, j'-1$. Constraint (8.27) works in a similar way and imposes that containers are relocated to adjacent tiers if they are put into the same stack. Constraint (8.28) limits the number of relocations to the upper bound per period $R^t_{\max}$. Constraints (8.29) and (8.30) define variable domains. We also fix variables x_{ijklnt} with the preprocessing step presented in Section 7.2.1.

Algorithm 8.2 describes how we use the binary subproblem within the column generation approach. We first run Heuristic HC and check, via the optimality criterion presented in section 6.3, if the heuristic solution is optimal. If yes, we return the optimal integer solution and do not solve the linear relaxed problem[1]. If not, column generation is started and the restricted master problem is initialized with columns from the heuristic solution and all bounds $R^t_{\max,1}$ are determined. At every iteration, the binary subproblem is run for each period $t = 1$ to $T-1$ and returns one column for each period. The process stops if none of these columns has negative reduced costs. In this case, we found an optimal linear solution for the restricted master problem.

[1]We are more interested in the optimal integer solution than in the optimal solution of the linear relaxation. In addition, we think that due to the preprocessing step the objective value of the linear relaxation is always equal or bigger than the lower bound LB.

Algorithm 8.2 Column generation with binary subproblem

```
run Heuristic HC
if optimality criterion holds then
  return optimal heuristic solution
end if
initialize restricted master problem (RMP)
compute all R^t_{max,1}
repeat
  add column(s) with negative reduced costs to RMP
  solve RMP
  for t = 1 to T − 1 do
    solve binary subproblem
  end for
until no column with negative reduced costs found
return optimal solution for RMP
```

We want to emphasize that not all columns added to the master problem may enter the solution immediately. The restricted master problem imposes consistency over time (e.g., no container can be retrieved or relocated from a position where it cannot be located or be relocated to a suspended position). Hence, it might happen that added columns violate consistency constraints if corresponding columns are not added to the model yet.

8.1.3 Information provided by dual variables

This section interprets information provided by dual variables and shows why they do not provide clear indications about the columns to be generated.

Variables δ_{ijt} represent the costs or benefits to retrieve container t from position (i, j), variables γ_{ijnt} the costs or benefits to take a relocation container $n > t$ from position (i, j), variables γ_{klnt} the costs or benefits to put a relocation container $n > t$ into position (k, l). Generally, $\delta_{ijt} < 0$ indicates that container t should be retrieved from position (i, j); $\gamma_{ijnt} < 0$ indicates that relocation container n should be taken out from position (i, j) and $\gamma_{klnt} > 0$ indicates that relocation container n should be put into position (k, l). To counterbalance the cost of 1 per relocation in the primal problem, $1 + \gamma_{ijnt} - \gamma_{klnt}$ should ideally be negative to relocate container n from position (i, j) to position (k, l).

Most of the time, dual variables do not provide such straightforward information. Most variables equal 0 and columns with negative reduced costs may have a retrieval or relocations with positive reduced costs ($\delta_{ijt} \geq 0$ or $1+\gamma_{ijnt}-\gamma_{klnt} \geq 0$). We illustrate different situations that lead to columns with negative reduced costs with an example. Figure 8.2 indicates values of dual variables at period 1 for a bay with 2 stacks and 3 tiers for retrieval container 1 and relocation containers 2, 3 and 4. We define c_{ijklnt} as the cost or benefit of a relocation: $c_{ijklnt} = 1 + \gamma_{ijnt} - \gamma_{klnt}$. Let c^- be the sum over all relocations of a column with $c_{ijklnt} < 0$ and c^+ the sum over all relocations of a column with $c_{ijklnt} > 0$.

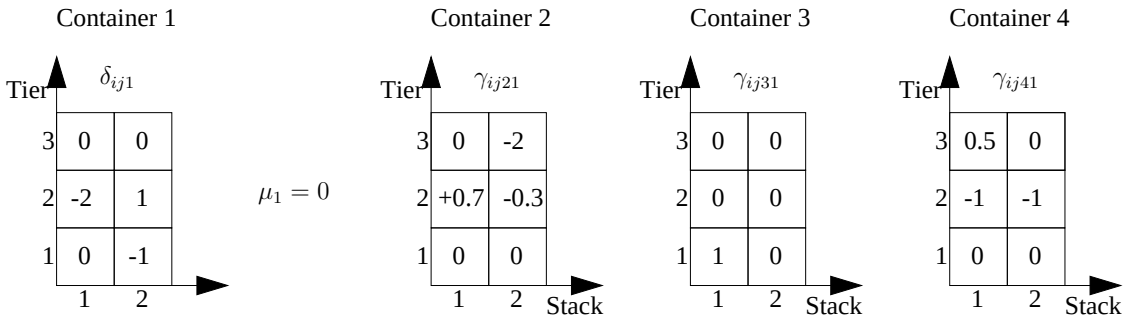


Figure 8.2: Dual variables for containers 1 to 4 for period 1

The following cases lead to columns with negative reduced costs:

- Beneficial retrieval and no relocation ($\delta_{ijt} < 0$):
  - column θ_1^1: $y_{1211}^1 = 1$ (reduced cost -2.0);
- Beneficial retrieval with beneficial relocation(s) ($\delta_{ijt} < 0$ and all $c_{ijklnt} < 0$):
  - column θ_1^2: $y_{2111}^2 = 1$ and $x_{221341}^2 = 1$ (reduced cost -1.5);
- Beneficial retrieval with costly relocation(s) ($\delta_{ijt} < 0$, all $c_{ijklnt} > 0$ and $\delta_{ijt} < -c^+$):
  - column θ_1^3: $y_{1211}^3 = 1$ and $x_{132121}^3 = 1$ (reduced cost -1.0);
- Costly retrieval with beneficial relocation(s) ($\delta_{ijt} \geq 0$, all $c_{ijklnt} < 0$ and $c^- < -\delta_{ijt}$):
  - column θ_1^4 with $y_{2211}^4 = 1$ and $x_{231221}^4 = 1$ (reduced cost -0.7);
- Neutral relocations ($c_{ijklnt} = 0$) that may be added to beneficial columns ($\delta_{ijt} + c^+ + c^- < 0$):
  - with $\gamma_{klnt} = 1$: column θ_1^5: y_{2111}^5 and $x_{221131}^5 = 1$ (reduced cost -1.0),
  - with $\gamma_{ijnt} = -1$: column θ_1^6: y_{2111}^6 and $x_{221141}^6 = 1$ (reduced cost -1.0),
  - with $\gamma_{ijnt} + \gamma_{klnt} = -1$: column θ_1^7: y_{2111}^7 and $x_{221221}^7 = 1$ (reduced cost -1.0);
- Costly relocations(s) ($c_{ijklnt} > 0$) that may be added to beneficial columns ($-c^+ < c^- + \delta_{ijt}$):
  - column θ_1^8: $y_{2111}^8 = 1$, $x_{221231}^8 = 1$ and $x_{231121}^8 = 1$ (reduced cost -1.0).

8.1.4 New bound on the number of relocations

We may use the values of dual variables to determine a new upper bound on the number of relocations per period t, $R^t_{\max,2}$. It uses the fact that the potential gain of a column has to outweigh its cost (the number of relocations). Equation (8.31) defines $R^t_{\max,2}$. This bound overestimates the maximum number of relocations, since it neglects several constraints (e.g., LIFO constraint, no holes, ...) that a column has to respect.

$$R^t_{\max,2} = \left\lceil \sum_{n=t+1}^{N} \left(\max\left(0, -\min_{\substack{i=1,\dots,I \\ j=2,\dots,J}} \gamma_{ijnt} + \max_{\substack{k=1,\dots,I \\ l=1,\dots,J}} \gamma_{klnt}\right)\right) - \min_{\substack{i=1,\dots,I \\ j=1,\dots,J}} \delta_{ijt} + \mu_t \right\rceil - 1$$
$$\forall t = 1, \dots, T-1 \tag{8.31}$$

Proof. Only columns with negative reduced costs are added to the restricted master problem. These columns satisfy Equation (8.32). Transforming this equation into Equation (8.33) shows that the number of relocations $\sum x^s_{ijklnt}$ is limited by the value of dual variables.

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt} \cdot (1 + \gamma_{ijnt} - \gamma_{klnt}) + \sum_{i=1}^{W}\sum_{j=1}^{H} y^s_{ijtt} \cdot \delta_{ijt} - \mu_t < 0 \tag{8.32}$$

$$\Leftrightarrow$$

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt} < \sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt} \cdot (-\gamma_{ijnt} + \gamma_{klnt}) - \sum_{i=1}^{W}\sum_{j=1}^{H} y^s_{ijtt} \cdot \delta_{ijt} + \mu_t \tag{8.33}$$

We are only interested in the maximum number of relocations and determine an upper bound on the right hand side (RHS) of Equation (8.33). The value of variable μ_t is given and cannot be influenced. To maximize RHS, we want to minimize $\sum y^s_{ijtt} \cdot \delta_{ijt}$. Each column executes exactly one retrieval and exactly one y^s_{ijtt} equals one. Hence,

$$\sum_{i=1}^{W}\sum_{j=1}^{H} y^s_{ijtt} \cdot \delta_{ijt} \geq \min_{\substack{i=1,\dots,I \\ j=1,\dots,J}} \delta_{ijt}$$

To maximize RHS, we want to maximize $\sum x^s_{ijklnt} \cdot (-\gamma_{ijnt} + \gamma_{klnt})$. Each container may be relocated exactly once and each relocation is composed of exactly one pick-up and one storage operation. For each container n, we obtain

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H} x^s_{ijklnt} \cdot (-\gamma_{ijnt} + \gamma_{klnt}) \leq -\min_{\substack{i=1,\dots,I \\ j=2,\dots,J}} \gamma_{ijnt} + \max_{\substack{k=1,\dots,I \\ l=1,\dots,J}} \gamma_{klnt}$$

A column may or may not contain relocations. Since we want to maximize RHS, we are only interested in relocations with $(-\gamma_{ijnt} + \gamma_{klnt} > 0)$. And obtain

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt} \cdot (-\gamma_{ijnt} + \gamma_{klnt}) \leq \sum_{n=t+1}^{N} \left(\max\left(0, -\min_{\substack{i=1,\dots,I \\ j=2,\dots,J}} \gamma_{ijnt} + \max_{\substack{i=1,\dots,I \\ j=1,\dots,J}} \gamma_{klnt}\right)\right)$$

We replace $\sum x^s_{ijklnt} \cdot (-\gamma_{ijnt} + \gamma_{klnt})$ and $\sum y_{ijtt} \cdot \delta_{ijt}$ in Equation (8.33) by their respective upper and lower bounds. We transform the fractional bound on the RHS into an integer value smaller than the bound and obtain Equation (8.31). □

8.2 Enumeration subproblem

This section presents two versions of an enumerative subproblem which enumerates all feasible columns. A feasible column retrieves the target container, relocates only blocking containers, respects the LIFO constraint and relocates containers to empty and non-suspended positions. We introduce two mechanisms to reduce the number of columns to enumerate.

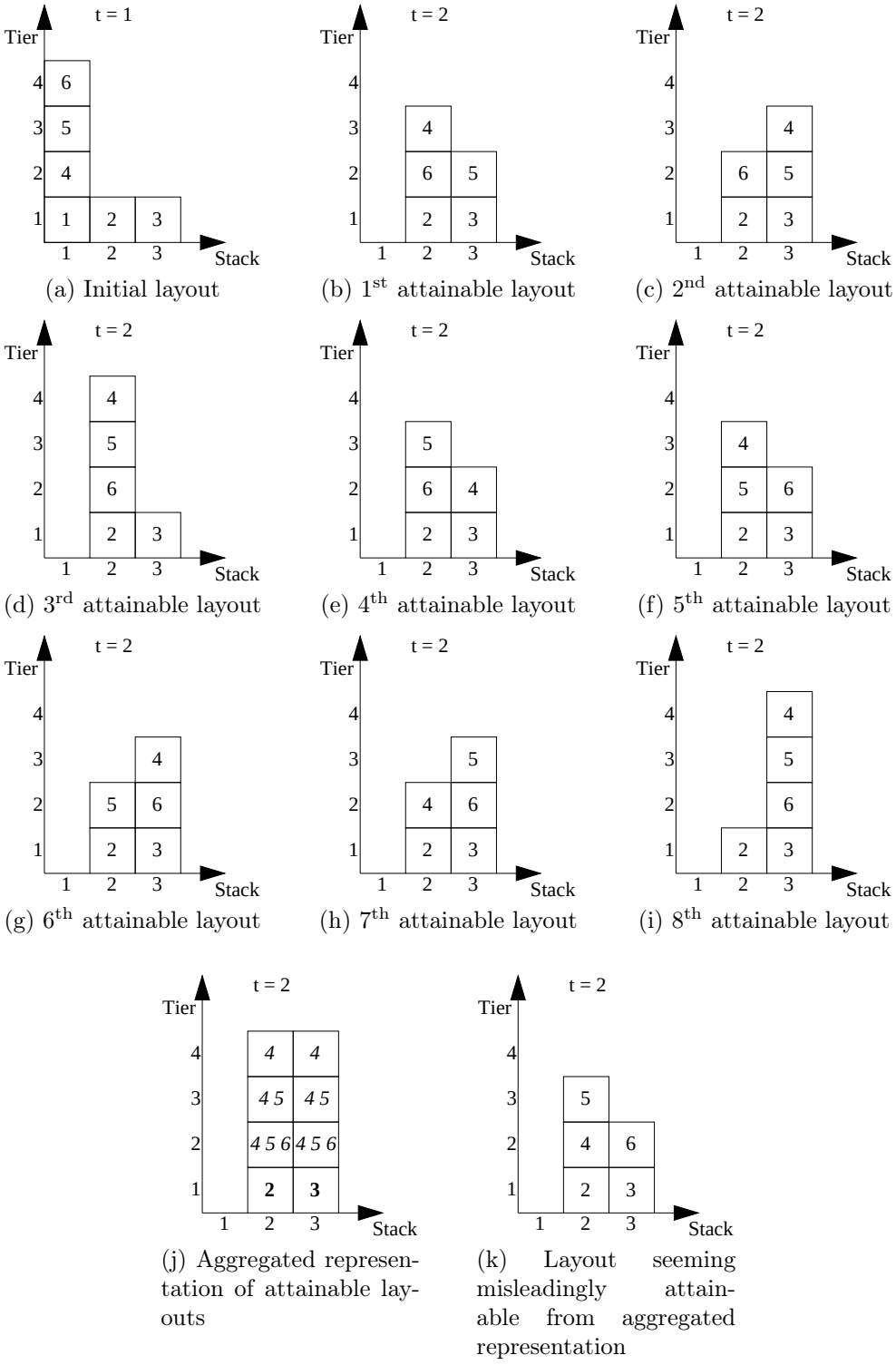
8.2.1 Enumeration based on attainable layouts

The initial bay layout restricts positions where containers may be located in subsequent periods. Attainable layouts are those layouts that can be reached from the initial layout via a series of feasible columns. Using information on possible container positions reduces the number of columns to enumerate at period t and hence the number of attainable layouts at period $t+1$. This also reduces the number of columns to enumerate at period $t+1$ and so on.

An example illustrates the benefit of attainable layouts. Figure 8.3a displays the initial configuration at period 1; Figures 8.3b to 8.3i all attainable layouts for period 2 (obtained by retrieving container 1 and relocating containers 4, 5 and 6). If we use the information about the layout at period 1, we generate only feasible columns that retrieve container 1 from position $(1,1)$, container 4 from position $(1,2)$, container 5 from position $(1,3)$ and container 6 from position $(1,4)$. Container 6 can only be relocated to positions $(2,1)$ and $(3,1)$; container 5 only to positions $(2,1)$, $(3,1)$, $(2,2)$ and $(3,2)$ (depending on the relocation of container 6); container 4 only to positions $(2,1)$, $(3,1)$, $(2,2)$, $(3,2)$, $(2,3)$ and $(3,3)$ (depending on the relocations of containers 5 and 6). Only eight columns are feasible and need to be enumerated. Without using the information on attainable layouts, all columns retrieving container 1 from every position $(1,1)$ to $(3,3)$ with all possible combinations of relocations of containers 2, 3 4, 5 and 6 have to be enumerated.

Since it is impractical to register all attainable layouts for each period, we use an aggregate way to do so. For each period t from 1 to $T-1$, we register which containers may be located at each position (i,j). This provides information about possible locations of all containers and possible empty positions. Figure 8.3j illustrates the aggregated representation for the example discussed above. A bold number indicates that the container is located at that position and nowhere else. An italic number indicates that the container may be located at that position or somewhere else. No entry indicates that the position is empty for sure. By aggregating all attainable layouts of the same period, we lose some information and unattainable layouts may seem attainable. Figure 8.3k illustrates such a layout.

We also use information on attainable layouts to tighten bound $R^t_{\max,2}$. In this case, $R^t_{\max,2}$ ignores dual variables that correspond to movements of containers from positions where containers cannot be located or to suspended or filled positions. Figure 8.4 illustrates the benefit. Figure 8.4a represents the values of dual variables for containers 1 to 9 at period 1. For each position (i,j), it displays the value of non-zero variables and the associated



(a) Initial layout

(b) 1st attainable layout

(c) 2nd attainable layout

(d) 3rd attainable layout

(e) 4th attainable layout

(f) 5th attainable layout

(g) 6th attainable layout

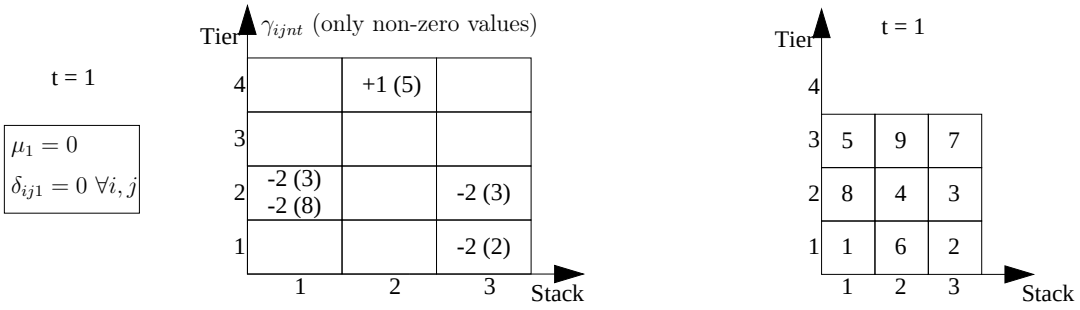
(h) 7th attainable layout

(i) 8th attainable layout

(j) Aggregated representation of attainable layouts

(k) Layout seeming misleadingly attainable from aggregated representation

Figure 8.3: Attainable layouts



(a) Values of dual variables and associated containers (in brackets) for $t = 1$

(b) Layout for $t = 1$

Figure 8.4: Computation of upper bound $R^t_{\max,2}$

container in brackets. All dual variables equal zero except $\gamma_{3,1,2,1} = -2$ for container 2, $\gamma_{1,2,3,1} = -2$ and $\gamma_{3,2,3,1} = -2$ for container 3, $\gamma_{2,4,5,1} = 1$ for container 5 and $\gamma_{1,2,8,1} = -2$ for container 8. Figure 8.4b represents the layout at period 1. Variables $\gamma_{1,2,3,1}$, $\gamma_{3,1,2,1}$ and $\gamma_{3,2,3,1}$ correspond to infeasible movements: either the container is not located at the position from which it should be retrieved or it is not placed above the target container and cannot be relocated. Only $\gamma_{1,2,8,1}$ and $\gamma_{2,4,5,1}$ are used to compute $R^1_{\max,2}$ and we obtain $R^1_{\max,2} = -(-2) + 1 - 1 = 2$. Without including information on attainable layouts we would have obtained $-(-2) + 1 - (-2) - 1 = 4$. The tightened bound may decrease the number of columns to enumerate significantly.

Algorithm 8.3 describes the column generation approach based on an enumerative subproblem using attainable layouts. Parts specific to this subproblem are colored. The solution procedure stops if the optimality criterion holds. Otherwise, we compute bounds $R^t_{\max,1}$ and generate attainable layouts for all periods $t = 1$ to $T - 1$. Attainable layouts at period t are determined by enumerating all feasible columns with at most $R^{t-1}_{\max,1}$ relocations for all attainable layouts at period $t-1$. We use the aggregated representation of attainable layouts throughout the entire solution procedure.

Then, the column generation process starts. The enumerative subproblem is called to search for columns with negative reduced costs. We use an upper bound on the number of relocations per period $R^t_{\max}$ to reduce the number of columns to be enumerated. $R^t_{\max}$ equals the minimum of $R^t_{\max,1}$ and $R^t_{\max,2}$. It is updated every time the subproblem is called since the value of $R^t_{\max,2}$ depends on the values of dual variables.

Columns with fewer relocations have higher chances to have negative reduced costs since each relocation has a cost of one. Therefore, the enumeration process starts enumerating columns with one relocation ($r = 1$) and iterates over all periods. We know the lower bound LB_t and the upper bound $R^t_{\max}$ on the number of relocations for feasible columns of period t with potentially negative reduced costs. The enumeration process is only executed for periods t where the number of relocations r is within these bounds. In this case, it enumerates all feasible columns at period t with r relocations. If no columns with negative reduced costs and exactly r relocations exist for any period t, the number of authorized relocations is increased by one.

The enumeration process stops if columns with negative reduced costs are found in any period t or if the maximum number of relocations is exceeded. The maximum number of

Algorithm 8.3 Column generation with enumerative subproblem

```
run Heuristic HC
if optimality criterion holds then
  return optimal heuristic solution
end if
initialize restricted master problem (RMP)
for t = 1 to T − 1 do
  compute R^t_max,1
  generate attainable layouts for t based on the aggregated
    representation of t − 1 and R^{t−1}_max,1
end for
repeat
  add column(s) with negative reduced costs to RMP
  solve RMP
  for t = 1 to T − 1 do
    compute R^t_max,2
    R^t_max ← min(R^t_max,1, R^t_max,2)
  end for
  for r = 1 to H − 1 do
    for t = 1 to T − 1 do
      if LB_t ≤ r and r ≤ R^t_max,1 then
        enumerate all feasible columns for t with exactly r relocations
        if column(s) with negative reduced costs exist then
          exit loops on r and t
        end if
      end if
    end for
  end for
until no column with negative reduced costs found
return optimal solution for RMP
```

relocations is limited by the height of the bay and set to $H-1$. In the first case, all columns of period t with negative reduced costs are added to the restricted master problem. The restricted master problem is solved and the enumerative subproblem is executed again. In the second case, an optimal linear solution is found and the column generation process ends.

8.2.2 Iterative solution approach

This section presents an iterative solution approach. It does not directly solve the entire problem to retrieve containers 1 to N. It starts, instead, by determining an optimal solution to retrieve container 1. Then, it determines an optimal solution to retrieve containers 1 and 2; then, an optimal solution to retrieve containers 1, 2 and 3; and so on. Like this, it iterates over all periods $t = 1$ to $T-1$. For each period t, it determines the optimal solution to retrieve containers 1 to t with a minimum number of relocations. To solve each of these problems, it uses the column generation approach with the enumerative subproblem described in the previous section.

Knowing the minimum number of relocations z_t needed to retrieve containers 1 to t makes

it possible to tighten $R^t_{\max,1}$ and to define a new optimality criterion. Constraint (6.4) recalls the definition of $R^t_{\max,1}$. It is computed with $\sum_{k=1}^{t-1} LB_k$ which provides a lower bound on the number of relocations to retrieve containers 1 to $t-1$. We replace the lower bound $\sum_{k=1}^{t-1} LB_k$ by the minimum number of relocations $\lceil z_{t-1} \rceil$. $R^t_{\max,1'}$ is defined by the resulting Constraint (8.34). Since $\lceil z_{t-1} \rceil \geq \sum_{k=1}^{t-1} LB_k$, $R^t_{\max,1'}$ is tighter than $R^t_{\max,1}$. It can easily be seen that the proof mechanism developed in Section 6.3 still holds.

$$R^t_{\max,1} = UB - 1 - \sum_{k=1}^{t-1} LB_k - \sum_{k=t+1}^{T} LB_k - \sum_{k=t}^{T} LB_{k+} \qquad \forall t = 1, \ldots, T \tag{6.4}$$

$$R^t_{\max,1'} = UB - 1 - \left\lceil z^{t-1} \right\rceil - \sum_{k=t+1}^{T} LB_k - \sum_{k=t}^{T} LB_{k+} \qquad \forall t = 1, \ldots, T \tag{8.34}$$

We also use $\lceil z^t \rceil$ to introduce a new optimality criterion: optimality criterion 2. If Constraint (8.35) holds for any period t, the lower bound equals the upper bound and the heuristic solution UB is optimal. The optimality criterion 2 is tighter than the one presented in Section 6.3 if and only if $\lceil z^t \rceil > \sum_{k=1}^{t} LB_k + \sum_{k=1}^{t-1} LB_{k+}$.

$$UB = \left\lceil z^t \right\rceil + \sum_{k=t+1}^{T} LB_k + \sum_{k=t}^{T} LB_{k+} \qquad \exists t = 1, \ldots, T \tag{8.35}$$

Proof. The optimality criterion 2 uses a new lower bound on the number of relocations to retrieve all containers from the bay. At least $\lceil z^t \rceil$ relocations are necessary to retrieve containers 1 to t. We use lower bounds LB_k and LB_{k+} introduced in Section 6.3 to compute a lower bound on the number of relocations to retrieve containers $t+1$ to T. Remember that LB_k relocations are executed in period k and LB_{k+} relocations in a period $k' > k$. Since we want to determine a lower bound on the number of relocations in periods $t+1$ to T, we suppose that all relocations LB_{k+} for $k = 1$ to $t-1$ are executed prior to period $t+1$. We obtain the following lower bound on periods $t+1$ to T: $\sum_{k=t+1}^{T} LB_k + \sum_{k=t}^{T} LB_{k+}$. Adding up $\lceil z^t \rceil$ and the lower bound on periods $t+1$ to T leads to Constraint (8.35).

Equation (6.3) recalls the optimality criterion presented in Section 6.3 in a slightly modified representation. Comparing Equations (8.35) and (6.3) shows that the optimality criterion 2 is tighter if and only if $\lceil z^t \rceil > \sum_{k=1}^{t} LB_k + \sum_{k=1}^{t-1} LB_{k+}$.

$$UB = \sum_{k=1}^{t} LB_k + \sum_{k=1}^{t-1} LB_{k+} + \sum_{k=t+1}^{T} LB_k + \sum_{k=t}^{T} LB_{k+} \tag{6.3}$$

□

Algorithm 8.4 describes the iterative column generation approach using an enumerative subproblem. It solves the problem to optimum for each period $t = 1$ to $T-1$. For each period t, it determines the optimal solution to retrieve containers 1 to t with the column generation approach described in Section 8.2.1. This approach uses an enumerative subproblem to generate columns and attainable layouts to reduce the number of columns to enumerate. Changes from the non-iterative approach are colored. The iterative approach uses the

tightened bound $R^t_{\max,1'}$ instead of $R^t_{\max,1}$ and checks after each iteration if the optimality criterion 2 holds. When solving the problem for period t, only columns for periods $t' = 1$ to t are generated since we are not interested in later periods.

Algorithm 8.4 Iterative column generation approach with enumerative subproblem

```
run Heuristic HC
if optimality criterion holds then
  return optimal heuristic solution
end if
initialize restricted master problem (RMP)
for t = 1 to T − 1 do
  compute R^t_{max,1'}
  generate all attainable layouts t based on the aggregated
    representation of t − 1 and R^{t−1}_{max,1'}
  repeat
    add column(s) with negative reduced costs to RMP
    solve RMP
    for t' = 1 to t do
      compute R^{t'}_{max,2}
      R^{t'}_{max} ← min(R^{t'}_{max,1'}, R^{t'}_{max,2})
    end for
    for r = 1 to H − 1 do
      for t' = 1 to t do
        if LB_{t'} ≤ r and r ≤ R^{t'}_{max} then
          enumerate all feasible columns for t with exactly r relocations
          if column(s) with negative reduced costs exist then
            exit loops on r and t'
          end if
        end if
      end for
    end for
  until no column with negative reduced costs found
  z^t ← solution value of RMP
  if optimality criterion 2 holds then
    return optimal heuristic solution
  end if
end for
return optimal solution for RMP
```

8.3 Branch and price

Column generation provides a linear relaxed solution of the integer problem. Branch and price embeds column generation in a branch and bound procedure to obtain the optimal integer solution. It uses column generation to solve the nodes in the branching tree. Algorithm 8.5 summarizes the general solution procedure. The root node is initialized with the restricted master problem and solved. If a fractional solution is obtained, new nodes are created by branching on fractional variables. Usually, variables of the initial problem are used for branching rather than columns. New nodes are added to the list of nodes to be handled. In the next iteration a node is retrieved from the list, solved via column generation and new nodes are created. The procedure stops if all nodes have been evaluated. In this case, the optimal integer solution has been determined.

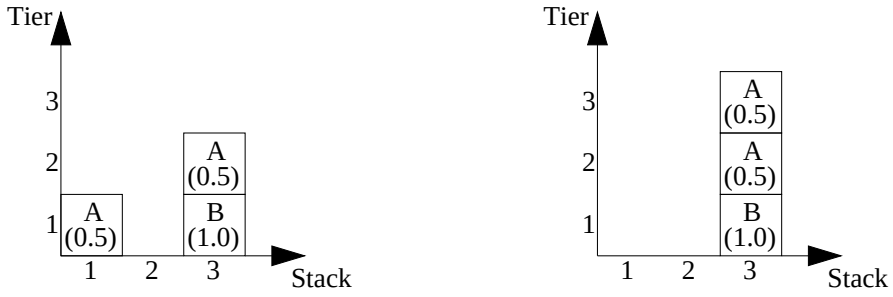
Algorithm 8.5 Branch and price procedure

```
rootNode ← initialize restricted master problem (RMP)
nodesToHandle ← rootNode
repeat
  node ← get node from nodesToHandle
  solve node with column generation
  create new nodes
  nodesToHandle ← new nodes
until nodesToHandle is empty
return optimal integer solution
```

In our case, containers may be split and occupy several positions within the bay for fractional solutions. Figure 8.5 illustrates two different situations that may occur: a container is put into different stacks or a container is put into different tiers of the same stack. We define the following branching strategies:

- If the container is split between two stacks i_1 and i_2, we create two nodes that forbid relocations to i_1 and i_2, respectively. One node prevents relocations to stack i_1 and to the first $\lceil W/2 \rceil$ stacks without stacks i_1 and i_2. The other one prevents relocations to stack i_2 and the remaining stacks.
- If the container is split between two tiers j_1 and j_2 of the same stack i_1, we create two nodes that forbid relocations to (i, j_1) and (i_1, j_2), respectively. The first node prevents relocations to tiers $1, ..., \min(j_1, j_2)$ of stack i_1. The other one prevents relocations to tiers $\min(j_1, j_2) + 1, ..., J$ of stack i_1.

We branch on variables b_{ijnt} (indicating if container n is placed at position (i, j) at the beginning of period t). If several split containers exist, we select the smallest period t' with a split container (fractional $b_{ijnt'}$). We determine the fractional variable $b_{ijnt'}$ whose value is closest to 0.5 and set i_1, j_1 and n' accordingly. If container n' is split between several stacks, we branch on stacks i_1 and the leftmost stack $i_2 \neq i_1$ containing container n'. If container n' is only split between different tiers in stack i_1, we branch on tier j_1 and the lowest tier $j_2 \neq j_1$ in stack i_1 containing container n'. To prevent that container n is located in position (i, j) at period t, we impose $b_{ijnt} = 0$ in the restricted master problem.



(a) Split into different stacks (b) Split into different tiers

Figure 8.5: Split container in a bay

8.4 Computational results

This section compares three column generation approaches against each other and the quality of their linear relaxations against the linear relaxation of a binary model. It also discusses the quality of the branching strategy. We compare the following approaches:

- Column generation with binary subproblem (sBIP);
- Column generation with enumerative subproblem (sEnum);
- Iterative column generation with enumerative subproblem (sIter);
- Branch and price (B&P) applied if the solution obtained by sIter is fractional and the time limit is not reached.

Experiments are carried out on a computer with Intel(R) Xeon(R) CPU clocked at 2.67GHz (dual core), 3.48GB RAM and operating with Windows XP Professional. We use Cplex 12.1 to solve the master problem and the binary subproblem. We limit the run time to 60 minutes per instance.

Table 8.1 shows the results of sBIP, sEnum and sIter in an aggregate way for non-trivial instances of sets 3-3 to 5-6. For sBIP, sEnum and sIter it indicates the number of instances that were solved to optimum and the number of instances that could not be solved because of time or memory limits. It also displays the average run time in seconds (solved and non-solved instances combined) and the percentage of time spent solving the master and the subproblem. For sEnum and sIter, it also shows the percentage of time needed to initialize attainable layouts. Instances 5-7 to 10-10 are not displayed since they cannot be solved due to memory problems (sBIP) or time limits (sEnum, sIter).

Results show that sBIP performs poorly: it solves only 42 of the smallest instances and runs out of time or memory for bigger instances. Closer analysis shows that around 80% of the solution time is needed to update the coefficients in the objective function for a huge number ($> W^2 \cdot H^2 \cdot N$) of variables. Consequently, almost no time is left to solve the subproblem and the master problem which explains the bad performance.

Approaches sEnum and sIter perform better: they solve all instances up to 4-4 and most instances up to 5-4. For bigger instances, they run mostly out of time. To initialize attainable layouts at period t, all feasible columns at period $t-1$ for all attainable layouts at period

Table 8.1: Performance comparison of column generation approaches sBIP, sEnum and sIter for non-trivial instances

	Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4	4-5	4-6	4-7	5-4	5-5	5-6	Total
	Nb. non-trivial	13	14	12	4	7	8	32	29	29	31	36	39	38	292
sBIP	Nb. solved	13	14	9	0	0	0	6	0	0	0	0	0	0	42
	Nb. time	0	0	3	4	7	8	25	28	28	0	24	0	0	127
	Nb. mem	27	26	28	36	33	32	9	12	12	40	16	40	40	351
	Avg. time [s]	25.9	417.2	2479.5	3625.6	3663.3	3904.4	3179.7	3525.9	3753.3	n.a.	3645.0	3882.6	n.a.	2918.4
	Time master [%]	0.8	0.3	0.2	0.1	0.1	0.2	3.5	3.5	3.6	n.a.	0.1	0.2	n.a.	1.2
	Time sub [%]	99.2	99.7	99.8	99.9	99.9	99.8	96.5	96.5	96.4	n.a.	99.9	99.8	n.a.	98.8
sEnum	Nb. solved	13	14	12	4	7	8	32	25	15	19	26	9	5	189
	Nb. time	0	0	0	0	0	0	0	4	14	10	10	30	33	101
	Nb. mem	0	0	0	0	0	0	0	0	0	2	0	0	0	2
	Avg. time [s]	0.0	0.2	0.3	884.2	93.7	21.9	28.2	1074.6	1892.5	1630.6	1534.5	2923.9	3454.0	1041.4
	Time layout [%]	0.0	12.6	27.1	59.9	61.7	48.5	46.8	62.0	76.3	66.4	59.8	93.7	95.2	54.6
	Time master [%]	90.2	74.5	66.1	30.2	28.8	44.4	33.1	23.7	19.3	20.1	19.2	1.1	0.2	37.5
	Time sub [%]	9.8	12.9	6.8	10.0	9.5	7.0	20.0	14.3	4.3	13.6	20.9	5.2	4.6	10.7
sIter	Nb. solved	13	14	12	4	7	8	32	28	26	22	25	10	6	207
	Nb. time	0	0	0	0	0	0	0	1	3	9	10	28	31	82
	Nb. mem	0	0	0	0	0	0	0	0	0	0	1	1	1	3
	Avg. time [s]	0.0	0.1	0.3	4.6	8.5	26.7	10.4	291.1	566.0	1346.1	1200.9	2725.4	3202.6	721.7
	Time layout [%]	0.0	5.9	14.9	45.0	49.8	36.6	16.5	37.7	50.0	57.3	35.2	77.0	86.1	39.4
	Time master [%]	96.8	82.4	69.6	34.0	33.9	46.7	56.5	31.9	32.6	23.0	30.0	2.3	1.9	41.7
	Time sub [%]	3.2	11.7	15.6	21.1	16.3	16.8	27.0	30.3	17.5	19.7	34.8	20.7	12.0	19.0
	Stop period	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	16.0	8.3	10.3	9.7	8.9	8.5	10.3

$t-1$ have to be enumerated. For bigger instances, the number of columns to enumerate explodes. This explains why - apart from the smallest instances - sEnum spends most of the time initializing attainable layouts. This implies that little time is used to actually solve the problem. For sIter, the problem is solved iteratively and attainable layouts are initialized iteratively. Due to the limited run time, for bigger instances the solution process stops far before executing the last iteration. Consequently, the percentage of time spent initializing attainable layouts is smaller but still significant.

Table 8.2 provides more information on sBIP, sEnum and sIter. It counts the number of instances for which an optimal integer or fractional solution is obtained. It indicates average values on all non-trivial instances (solved and non-solved) for the number of times the subproblem was called, the number of generated columns and the number of columns added to the master problem. For sEnum and sIter, it displays average values of feasible columns which represent the number of columns that are enumerated while initializing attainable layouts.

Approach sBIP obtains mostly fractional solutions. The subproblem is rarely called and few columns are generated. We already mentioned that most of the solution time is spent updating the objective function of the subproblem which leaves little time to solve the master and the subproblem. In addition, the subproblem is called for each period $t = 1$ to $T-1$ and generates exactly one column per period. Around 40% to 70% of the generated columns are added to the model. This implies that the subproblem is frequently solved to optimum only to discover that no column with negative reduced costs exists for the given period.

Approach sEnum obtains more integer solutions than sBIP. It generates only columns that may be applied to attainable layouts whereas sBIP does not use this restriction. Results show that using attainable layouts to create fewer columns leads to a better solution quality. For sEnum, the subproblem is called rarely. For small instances, few iterations are necessary to obtain the optimal solution; for big instances, most of the time is spent initializing attainable layouts and few time solving the problem. The huge number of feasible and generated columns - hundreds of millions for some instances - shows how the solution space explodes. This illustrates the limits of an enumerative solution approach where all feasible columns are enumerated every time the subproblem is called. In addition, at most 4% of generated columns are added to the master problem.

The table distinguishes the number of columns that are used to initialize attainable layouts (feasible columns) and the number of columns enumerated while executing the subproblem (generated columns). The number of generated columns can be lower than the number of feasible columns for two reasons: first, the bound $R^t_{\max,2}$ provided by dual variables might be tighter than bound $R^t_{\max,1}$ and fewer columns have to be enumerated; second, all time is spent initializing attainable layouts.

Approach sIter obtains more integer solutions than sEnum. The reason is the additional optimality criterion 2 which proves the heuristic solution to be optimal. For sIter, the subproblem is executed more often since the problem is solved for each iteration. However, tightened bounds and optimality criterion 2 reduce the number of feasible and generated columns and reduce the overall solution time.

Table 8.2: Details on performance of column generation approaches sBIP, sEnum and sIter for non-trivial instances

	Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4
	Nb. non-trivial	13	14	12	4	7	8	32
sBIP	Nb. int sol	2	4	1	0	0	0	3
	Nb. frac sol	11	10	8	0	0	0	3
	Iter. sub	49.8	100.1	127.0	53.5	19.3	8.0	128.8
	Nb. added col	192.3	549.4	867.9	675.8	118.9	58.4	1 281.8
	Nb. gen col	398.8	1 101.6	1 778.0	909.5	385.7	184.0	1 994.0
sEnum	Nb. int sol	5	3	1	0	0	0	7
	Nb. frac sol	8	11	11	4	7	8	25
	Iter. sub	12.5	19.4	18.7	39.8	37.6	36.0	44.8
	Nb. added col	86.6	458.5	507.7	3 114.3	8 256.7	9 250.6	9 642.0
	Nb. gen col	1 166.0	29 825.9	12 417.4	209 719.5	1 377 823.4	1 043 005.6	2 633 877.5
	Nb. feas col	451.8	23 303.5	66 781.5	440 756 904.8	32 290 233.3	6 862 157.3	5 749 830.1
sIter	Nb. int sol	11	10	10	1	5	4	12
	Nb. frac sol	2	4	2	3	2	4	20
	Iter. sub	16.7	23.5	28.3	50.3	44.9	44.0	45.9
	Nb. added col	76.8	251.4	521.3	5 570.3	7 130.7	5 580.4	3 250.0
	Nb. gen col	1 070.9	10 540.3	22 025.8	356 901.5	695 383.4	3 550 736.9	2 336 368.3
	Nb. feas col	169.9	3 340.0	30 970.1	1 025 303.3	1 924 570.9	4 961 504.8	624 293.3

	Inst. set	4-5	4-6	4-7	5-4	5-5	5-6	Total
	Nb. non-trivial	29	29	31	36	39	38	292
sBIP	Nb. int sol	0	0	0	0	0	0	10
	Nb. frac sol	0	0	0	0	0	0	32
	Iter. sub	34.8	10.6	n.a.	40.3	9.9	n.a.	52.9
	Nb. added col	376.3	99.7	n.a.	511.8	142.0	n.a.	443.1
	Nb. gen col	684.0	252.2	n.a.	764.8	238.2	n.a.	790.1
sEnum	Nb. int sol	6	2	2	4	0	2	32
	Nb. frac sol	19	13	17	22	9	3	157
	Iter. sub	44.8	21.0	49.8	64.9	18.2	17.3	32.7
	Nb. added col	10 262.1	10 371.1	14 913.5	16 963.1	5 972.3	7 632.7	7 494.7
	Nb. gen col	36 142 781.6	1 006 292.6	29 600 523.8	66 498 991.9	4 966 243.4	52 674 744.8	15 092 108.7
	Nb. feas col	322 584 957.3	460 660 298.8	350 540 653.3	239 502 368.6	395 992 607.9	406 416 667.3	204 726 708.9
sIter	Nb. int sol	11	8	11	6	4	0	93
	Nb. frac sol	17	18	11	19	6	6	114
	Iter. sub	58.0	60.6	55.0	70.1	40.9	41.7	44.6
	Nb. added col	23 994.0	57 122.3	73 683.7	24 327.3	24 198.8	44 079.3	20 752.8
	Nb. gen col	59 039 229.8	13 296 863.4	29 482 965.5	108 459 993.5	58 488 032.0	53 212 276.5	25 304 029.8
	Nb. feas col	34 914 576.5	151 431 035.8	292 892 757.8	216 009 318.4	446 664 254.6	456 150 375.1	123 587 113.1

Table 8.3: Performance comparison of iterative column generation and BIP 2

	Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4
	Nb. non-trivial	13	14	12	4	7	8	32
sIter	Nb. solved	13	14	12	4	7	8	32
	Avg. time [s]	0.0	0.1	0.3	4.6	8.5	26.7	10.4
	LB improved	4	5	7	3	5	7	13
BIP2	Nb. solved	13	14	12	4	7	8	32
	Avg. time [s]	0.1	0.4	1.2	4.5	6.6	11.2	2.0

	Inst. set	4-5	4-6	4-7	5-4	5-5	5-6	Total
	Nb. non-trivial	29	29	31	36	39	38	292
sIter	25	15	19	26	9	5	189	
	Avg. time [s]	171.8	215.8	423.5	239.8	273.0	1134.1	193.0
	LB improved	15	10	13	16	3	3	104
BIP2	Nb. solved	29	27	24	36	30	12	248
	Avg. time [s]	15.4	24.6	84.4	329.3	356.5	828.9	128.1

Table 8.3 compares the results of sIter to the results obtained with the binary model BIP2 and its linear relaxation LR_BIP2 from Section 7.3. It indicates the number of instances solved by sIter and BIP2 and the average solution times of solved instances. It also displays how often sIter tightens the lower bound obtained by LR_BIP2. We see that sIter provides tighter lower bounds than LR_BIP2 for 104 instances. However, BIP2 has similar run times than sIter, solves more instances and provides directly the integer solution. Consequently, it seems not beneficial to apply the column generation approach since small instances can directly be solved by the binary programming model and bigger instances cannot be solved by the column generation approach within reasonable time due to the huge number of columns to be enumerated.

We apply a branch and price approach on instances where sIter obtains a fractional solution and does not reach the time limit. We use sIter to solve the root node, sEnum to solve nodes in the branching tree and the branching procedure described in Section 8.3. Table 8.4 presents the results. It indicates the number of instances for which branch and price was executed. It displays how often the branch and price process finished and how often it was stopped due to the time limit of one hour. For both cases, the table presents the number of solved nodes and the average solution time per node.

Results show that branch and price finds the optimal integer solution for some smaller instances. But for most instances, especially for bigger ones, it reaches the time limit. For solved instances, the optimal solution value equals the round up value of the fractional solution at the root node in these cases. This explains why relatively few nodes have to be explored to find the optimal solution. If the time limit is reached either thousands of nodes have to be explored or the solution time per node takes several minutes. Results displayed in this table are obtained with a width-first branching strategy which obtained better results than a best-node-first strategy. Probably, only the width-first strategy finds

Table 8.4: Performance of branch and price combined with sIter for non-trivial instances

Inst. set	3-3	3-4	3-5	3-6	3-7	3-8	4-4
Nb. Inst	2	4	2	3	2	4	20
Nb. finished	1	4	1	1	1	1	12
Nb. nodes	1.0	8.0	41.0	1.0	317.0	109.0	344.3
Avg. time node [s]	0.5	0.0	0.0	0.1	0.3	0.0	0.6
Nb. not finished	1	0	1	2	1	3	8
Nb. nodes	n.a	n.a.	151 614.0	1 277.5	318.0	894.0	25 270.8
Avg. time node [s]	n.a	n.a.	0.0	8.5	11.3	11.2	21.2

Inst. set	4-5	4-6	4-7	5-4	5-5	5-6	Total
Nb. Inst	17	18	11	19	6	6	114
Nb. finished	1	0	0	2	0	0	24
Nb. nodes	33.5	4.0	n.a.	19.0	n.a.	n.a.	87.8
Avg. time node [s]	3.6	1.6	n.a.	0.8	n.a.	n.a.	0.8
Nb. not finished	16	18	11	17	6	6	90
Nb. nodes	151.9	2 861.4	109.0	6 429.0	69.6	24.5	17 183.6
Avg. time node [s]	80.3	121.8	199.6	52.6	446.6	206.4	105.4

an integer solution close to the initial fractional solution. Both strategies run out of time if the gap between the solution at the root node and the optimal integer solution is bigger.

8.5 Conclusion

This chapter introduced the first column generation approach for the container relocation problem. We defined columns representing the retrieval of one container and relocations of any blocking containers. We presented a decomposition to split the initial binary programming model into a master problem - ensuring the bay consistency over time - and a binary subproblem - generating new feasible columns. We introduced a new upper bound on the number of relocations per period based on the values of dual variables.

In addition, to the binary subproblem we also presented two implementations of an enumerative subproblem to generate feasible columns. We introduced 'attainable layouts' to indicate if a layout may be reached from the initial layout. We use attainable layouts to reduce the number of columns to be enumerated. The second variant, solves the problem iteratively for each period $t = 1$ to $T - 1$. This makes it possible to tighten the bound on the maximum number of relocations per period and to introduce a new optimality criterion to check if the heuristic solution is optimal.

Results show that column generation with the binary subproblem performs poorly. Most of the solution time is used to update the objective function of the subproblem due to the huge number of variables x_{ijklnt}. Maybe a binary subproblem where variables x_{ijklnt} are split into variables x_{ijnt} for retrieval and z_{klnt} for storage could lead to better results. This would

reduce the number of variables from $W^2 \cdot H^2 \cdot N$ to $2 \cdot W \cdot H \cdot N$, but would require some new constraints.

Both enumerative subproblems perform well for smaller instances, but are impractical for bigger instances due to the huge number of columns (up to hundreds of millions) to enumerate. Consequently, a subproblem which does not need to enumerate all feasible columns has to be designed to be able to solve bigger instances.

We also introduced a branching procedure that branches on variables indicating container positions. It has been applied together with the enumerative subproblem in a branch and price approach. The branching procedure works fine for problems where the fractional solution is very close to the optimal value, but does not obtain the optimal value otherwise. This is due to the fact that a lot of time is needed to solve nodes in the branching tree because of the long run times of the enumerative subproblem. Before improving the branch and price approach, the column generation has to be sped up first.

Even if the column generation approach does not work as well as expected, it helped to get a deeper understanding of the problem. This enabled us to improve the binary formulation and to introduce the preprocessing step described earlier and the heuristic approach described in the next chapter.

Chapter 9

Heuristic branch and price approach for CRP

The previous chapter showed the limits of an exact branch and price approach using an enumerative subproblem. This approach is impractical due to the huge number of columns that have to be enumerated. This chapter presents a heuristic branch and price approach that overcomes this problem. It uses the same master problem (see Section 8.1.1) and a heuristic subproblem. This heuristic approach is meant to determine a good integer solution rather than an optimal fractional solution.

Section 9.1 presents the heuristic subproblem that generates columns by solving a network flow problems and applying heuristic relocation rules. Section 9.2 describes the heuristic branch and price approach. It details the branching procedure and shows how heuristic HC is used to obtain integer solutions. Section 9.3 presents computational results. Section 9.4 concludes the chapter.

9.1 Heuristic subproblem

This section presents a heuristic pricing subproblem to determine columns to be added to the master problem. It is twofold: first, it decides which relocation containers should be picked up; second, it decides where to relocate these containers. These decisions are taken based on the values of dual variables and heuristic relocation rules. As before, we use attainable layouts and different bounds to speed up the subproblem.

Algorithm 9.1 summarizes the column generation approach with the heuristic subproblem. The restricted master problem is initialized with columns corresponding to the solution of Heuristic HC. Attainable container positions describe at which positions each container may be located at each period. Attainable positions at period 1 are identical to the initial layout. For later periods, attainable positions are updated with relocations of columns added to the restricted master problem.

During the solution procedure, the subproblem is executed for each period t. It generates columns for every position (i, j) where the retrieval container t may be located. For each position, we determine an upper bound on the number of relocations needed to retrieve the target container. For this purpose, we use attainable container positions. We count the number of potentially filled positions and the number of different containers above the

Algorithm 9.1 Column generation with heuristic subproblem

```
initialize restricted master problem (RMP) and attainable positions
repeat
  add column(s) with negative reduced cost to RMP
  update attainable positions with added column(s)
  solve RMP
  for t = 1 to T − 1 do
    for all attainable positions (i, j) of retrieval container t do
      compute R_(i,j) based on attainable container positions
      compute stack heights s_k^min and s_k^max
      for r = LB_t to R = min(R^t_max,1, R_(i,j)) do
        if columns with r relocations may have negative reduced costs then
          determine pick-up of r relocation containers (step 1)
          update stack heights s_k^max for pick ups
          if columns with given pick-ups may have negative reduced costs then
            determine input of r relocation containers (step 2)
          end if
        end if
      end for
    end for
  end for
until no column with negative reduced costs found
return solution for RMP
```

target container. The bound $R_{(i,j)}$ is set to the minimum of these two values. The maximum number of relocations R equals the minimum of $R^t_{\max,1}$ and $R_{(i,j)}$.

We illustrate the computation of $R_{(i,j)}$ with two examples. Figure 9.1 displays an initial bay layout with 16 containers. Figure 9.2 shows attainable positions for containers 9 to 16 at period 9 for this layout. First, we compute $R_{(4,3)}$ to retrieve container 9 from position $(4,3)$ (colored). Attainable container positions indicate that positions $(4,4)$ to $(4,6)$ may be occupied. They also show that containers 11, 12, 13 and 15 may be located in these three positions. We obtain $R_{(4,3)} = \min\{3,4\} = 3$. Second, we determine $R_{(4,1)}$ to retrieve container 9 from position $(4,1)$ (colored). Positions $(4,2)$ to $(4,6)$ may be occupied. Containers 11, 12, 13 and 15 may be located in these five positions and $R_{(4,1)} = \min\{5,4\} = 4$.

We aim to generate only columns that can be applied in an integer solution. We try not to generate columns that relocate containers to suspended positions. For this purpose, we determine minimum and maximum stack heights at period t, $s_k^{\min}$ and $s_k^{\max}$, for all stacks k. We allow only relocations to positions (k,l) with $s_k^{\min}+1 \le l \le s_k^{\max}+r$. We use the initial bay layout to determine $s_k^{\min}$. We simply count the number of containers that are neither retrieved nor relocated in stack k up to period t (t included). Constraint (9.1) defines $s_k^{\max}$ based on values of variables b_{ijnt} of the restricted master problem. The current retrieval container is excluded since it is impossible to relocate containers on its top.

$$s_k^{\max} = \left\lceil \sum_{n=t+1}^{N} \sum_{l=1}^{H} b_{klnt} \right\rceil \qquad \forall k = 1 \ldots, W \tag{9.1}$$

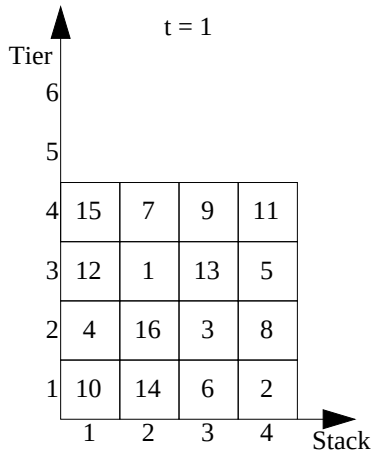


Figure 9.1: Initial bay layout at period 1

6	9 11 12 13 15	9 11 12 13 15		9 11 12 13 15
5	9 11 12 13 15	9 11 12 13 15	9 11 12 13 15	9 11 12 13 15
4	9 11 12 13 15	9 11 12 13 15	9 11 12 13 15	9 11 12 13 15
3	9 11 12 13 15	9 11 12 13 15	9 11 12 13 15	**9** 11 12 13 15
2	9 11 12 13 15	16	9 11 12 13 15	9 11 12 13 15
1	10	14	9 11 12 13 15	**9** 11 12 13 15
	1	2	3	4

Figure 9.2: Attainable container positions at period 9

6		9 (0.13)		
5		9 (0.35)		12 (0.39)
4		9 (0.13) 11 (0.22)		12 (0.19) 13 (0.17) 15 (0.03)
3		11 (0.78)		9 (0.03) 12 (0.22) 13 (0.17)
2		16 (1.00)		9 (0.14) 12 (0.20) 13 (0.22) 15 (0.22)
1	10 (1.00)	14 (1.00)	15 (0.54)	9 (0.22) 13 (0.43) 15 (0.22)
	1	2	3	4
$s_k^{\min}$	1	2	0	0
$s_k^{\max}$	1	3	1	3

Figure 9.3: Bay layout at period 9 indicating container positions and values of variables $b_{ijnt} > 0$ (values in brackets) as well as minimum and maximum stack heights

We illustrate the computation of $s_k^{\max}$ with the help of Figure 9.3. It represents the bay configuration at period 9 given by values of variables b_{ijnt}. Only variables $b_{ijnt} > 0$ are shown. We compute $s_k^{\max}$ with Constraint (9.1) for this configuration and obtain: $s_1^{\max} = \lceil 1.00 \rceil = 1$, $s_2^{\max} = \lceil 3.00 \rceil = 3$, $s_3^{\max} = \lceil 0.54 \rceil = 1$ and $s_4^{\max} = \lceil 2.46 \rceil = 3$. We determine $s_k^{\min}$ for the initial layout depicted in Figure 9.1 and obtain: $s_1^{\min} = 1$ (container 10), $s_2^{\min} = 2$ (containers 14 and 16), $s_3^{\min} = 0$ and $s_4^{\min} = 0$.

We use the information provided by R, $s_k^{\min}$ and $s_k^{\max}$ during the next steps. For each retrieval position (i, j), we create columns with different numbers of relocations. The number of relocations is bounded by the initial lower bound LB_t and the maximum number of relocations R. We generate columns for each tuple (t', i', j', r') to retrieve container t' from position (i', j') and to relocate r' blocking containers. We also impose that containers are only relocated to positions (k, l) with $k \neq i'$ and $s_k^{\min} + 1 \leq l \leq s_k^{\max} + r'$.

We introduce two lower bounds on negative reduced costs to check if columns with

negative reduced costs may exist for tuple (t', i', j', r'). We use the following notation to formulate the bounds:

- $\mathcal{N}^j$ contains all containers that may be located at position (i', j);
- γ_n^{in} represents the maximum value to put relocation container n into position (k, l): $\gamma_n^{in} = \max_{(k=1,\ldots,i'-1,i'+1,\ldots,W,l=s_k^{\min}+1,\ldots,s_k^{\max}+r')}(\gamma_{klnt'})$;
- τ_n represents the maximum benefit that can be obtained by relocating container n: $\tau_n = \min_{(j=j'+1,\ldots,j'+r'|n\in\mathcal{N}^j)}(\gamma_{i'jnt'} - \gamma_n^{in})$;
- $\mathcal{N}^r$ is a set of size r' that contains containers with the smallest τ_n;
- τ_{ij} represents the maximum benefit that can be obtained by relocating containers from position (i, j): $\tau_{ij} = \min_{(n\in\mathcal{N}^j)}(\gamma_{i'jnt'} - \gamma_n^{in})$.

Constraint (9.2) determines a lower bound on reduced costs via the maximum benefit of containers; Constraint (9.3) via the maximum benefit of positions. Only if both constraints hold, the heuristic subproblem may generate columns with negative reduced costs.

$$r' + \sum_{n\in\mathcal{N}^{r'}} \tau_n + \delta_{i'j't'} - \mu_{t'} < 0 \tag{9.2}$$

$$r' + \sum_{j=j'+1}^{j'+r} \tau_{i'j} + \delta_{i'j't'} - \mu_{t'} < 0 \tag{9.3}$$

Proof. Only columns with negative reduced costs are added to the restricted master problem. Bounds are computed for a given tuple (t', i', j', r'). This implies that container t' is retrieved from position (i', j') and r' containers are relocated from positions $(i', j'+1)$ to $(i', j'+r)$ to positions (k, l) with $k \neq i'$ and $s_k^{\min} + 1 \leq l \leq s_k^{\max} + r'$. In addition, containers may only be picked up from attainable positions. Equation (9.4) defines reduced cost for this specific case. It is an adapted version of the general Equation (8.32).

$$\sum_{j=j'+1}^{j'+r'} \sum_{k\neq i'} \sum_{l=s_k^{\min}+1}^{s_k^{\max}+r'} \sum_{n\in\mathcal{N}^j} x^s_{i'jklnt'} \cdot (1 + \gamma_{i'jnt'} - \gamma_{klnt'}) + \delta_{i'j't'} - \mu_{t'} < 0 \tag{9.4}$$

We know that $\sum\sum\sum\sum x^s_{i'jklnt'} \cdot 1 = r'$. The term $\sum_{n\in\mathcal{N}^{r'}} \tau_n$ is a valid lower bound on $\sum\sum\sum\sum x^s_{i'jklnt'} \cdot (\gamma_{i'jnt'} - \gamma_{klnt'})$ since it is computed with the most beneficial containers. The term $\sum_{j=j'+1}^{j'+r} \tau_{i'j}$ is a valid lower bound on $\sum\sum\sum\sum x^s_{i'jklnt'} \cdot (\gamma_{i'jnt'} - \gamma_{klnt'})$ since it is computed with the highest benefit for each position. □

We illustrate the computation of γ_n^{in}, τ_n and τ_{ij} with the help of Figure 9.4. It presents non-zero values of dual variables γ_{ijnt} at period 9. We consider the case where container 9 has to be retrieved from position $(4, 3)$ with three relocations: $t' = 9$, $i' = 4$, $j' = 3$ and $r' = 3$. This means that containers can be picked up from positions $(4, 4)$ to $(4, 6)$. These positions are colored with dark gray. We determine possible relocation containers for these

	1	2	3	4
6				-0.61 (12)
5		0.13 (12)	0.84 (12)	0.13 (12) -0.63(13) -0.35 (15)
4		0.17 (10) 0.26 (12) 0.61 (13) 0.17 (14) 0.17 (16)		0.13 (12) -0.63 (13) -0.17 (14) -0.35 (15)
3		0.39 (10) 0.39 (12) 1.00 (13) 0.39 (14) 0.39 (15) 0.39 (16)	0.39 (12) 0.98 (13)	0.35 (10) 0.35 (11) 0.13 (12) 0.98 (13) -0.17 (15)
2	0.09 (12) 0.60 (15)		0.39 (12) 0.98 (13) 0.65 (15)	-0.30 (10) -0.30 (11) -0.17 (12) 0.50 (13) -0.30 (14) -0.30 (16)
1			0.39 (12) 0.98 (13) 0.65 (15)	0.35 (11) 0.54 (12) 0.80 (13) 1.00 (15) 1.20 (16)

$\delta_{439} = 0.33$	$\mu_9 = 0.33$		
$\gamma^{in}_{11} = 0.00$	$\gamma^{in}_{12} = 0.39$	$\gamma^{in}_{13} = 1.00$	$\gamma^{in}_{15} = 0.65$
$\tau_{11} = 0.00$	$\tau_{12} = -1.00$	$\tau_{13} = -1.63$	$\tau_{15} = -1.00$
$\tau_{4,4} = -1.63$	$\tau_{4,5} = -1.63$	$\tau_{4,6} = -1.00$	

Figure 9.4: Values of dual variables γ_{ijnt} at period 9 with associated containers (in brackets) and resulting benefits for relocations

positions with Figure 9.2: $\mathcal{N}^4 = \mathcal{N}^5 = \mathcal{N}^6 = \{11, 12, 13, 15\}$. Positions to which containers may be relocated depend on minimum and maximum stack heights (see Figure 9.3). Possible positions are colored with light gray.

We illustrate the computation of γ^{in}_n, τ_n and τ_{ij} on some examples: $\gamma^{in}_{12} = \max(0.00, 0.09, 0.13, 0.26, 0.39) = 0.39$, $\tau_{13} = \min(-0.63 - 1.00, -0.63 - 1.00, 0.00 - 1.00) = -1.63$ and $\tau_{4,4} = \min(0.00 - 0.00, 0.13 - 0.39, -0.63 - 1.00, -0.35 - 0.65) = -1.63$. All other parameters are determined in the same way. The resulting values are reported in Figure 9.4. From the dual solution we obtain $\delta_{439} = 0.33$ and $\mu_9 = 0.33$. Here, columns with negative reduced costs may exist: Constraint (9.2) holds since $3 + (-1.63 - 1.00 - 1.00) + 0.33 - 0.33 = -0.63$; Constraint (9.3) holds since $3 + (-1.63 - 1.63 - 1.00) + 0.33 - 0.33 = -1.26$.

Step 1 - Pickup of relocation containers

If columns with negative reduced may be generated for tuple (t', i', j', r'), we determine which container n is picked up from each position $(i', j' + 1)$ to $(i', j' + r)$. We use a minimum cost flow problem to represent possible pick-ups and their benefits. We define a directed graph $G = (V, A)$ with vertex set $V = V1 \cup V2 \cup V3 \cup V4$ and arc set $A = A1 \cup A2 \cup A3$. We also define lower and upper bounds l and u on arcs $A1$, $A2$ and $A3$ and weights k on arcs $A2$.

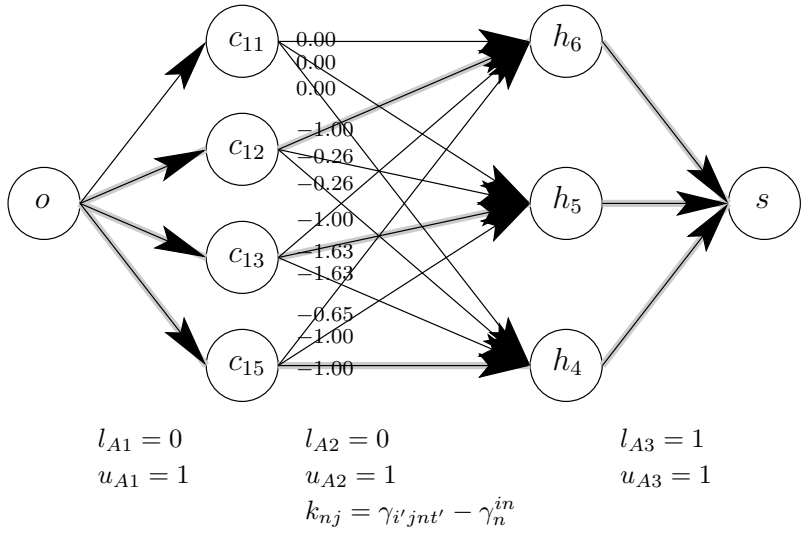


Figure 9.5: Minimum cost flow network to determine container pick-ups for $y_{4,3,9,9}$ and $r = 3$ and its optimal solution

- $V1 = \{o\}$ where o is the source node;
- $V2 = \{c_n \mid n \in \mathcal{N}^j \; \exists \; j = j' + 1, \ldots, j' + r'\}$ where c_n represents container n;
- $V3 = \{h_j \mid j' + 1 \leq j \leq j' + r'\}$ where h_j represents position (i', j);
- $V4 = \{s\}$ where s is the sink node;
- $A1 = \{(o, c_n)\}$
  $l_{A1} = 0$ and $u_{A1} = 1$ since a container may or may not be picked up;
- $A2 = \{(c_n, h_j) \mid n \in \mathcal{N}^j\}$
  $l_{A2} = 0$ and $u_{A2} = 1$ since a container may or may not be picked up,
  $k_{nj} = \gamma_{i'jnt'} - \gamma_n^{in}$: maximum benefit to pick up container n from position (i', j) and relocate it;
- $A3 = \{(h_j, s)\}$
  $l_{A3} = u_{A3} = 1$ since a container has to be picked up from position (i', j).

Figure 9.5 illustrates the flow network for our example where container 9 has to be retrieved from position $(4, 3)$ and where three blocking containers have to be relocated. It also illustrates an optimal solution for this problem where container 12 is picked up from position $(4, 6)$, container 13 from position $(4, 5)$ and container 15 from position $(4, 4)$. The minimum cost is -3.63.

We implement a linear programming model to solve the above described minimum cost flow problem. Linear variables in_n describe flows on arcs $A1$, variables a_{nj} flows on arcs $A2$ and parameters out_j flows on arcs $A3$. We set $in_n = 0$ if arc (o, c_n) does not exist, $a_{nj} = 0$ if arc (c_n, h_j) does not exist, $out_n = 0$ if arc (h_j, s) does not exist and $out_n = 1$ if arc (h_j, s) does exist. We obtain

$$\min \sum_{n=1}^{N} \sum_{j=1}^{H} k_{nj} \cdot a_{nj}$$

s.t.

$$in_n = \sum_{j=1}^{H} a_{nj} \qquad \forall n = 1, \ldots, N \tag{9.5}$$

$$\sum_{n=1}^{N} a_{nj} = out_j \qquad \forall j = 1, \ldots, H \tag{9.6}$$

$$0 \leq in_n \leq 1 \qquad \forall n = 1, \ldots, N \tag{9.7}$$

$$0 \leq a_{nj} \leq 1 \qquad \forall n = 1, \ldots, N, j = 1, \ldots, H \tag{9.8}$$

The objective function minimizes the cost of the flow. Constraints (9.5) and (9.6) impose mass balance constraints. Constraints (9.7) and (9.8) define variable domains. Since we solve a flow problem all variables a_{nj} take integer values. The solution of this flow problem indicates which containers have to be picked up: $a_{nj} = 1$ indicates that container n should be picked up from position (i', j). The set $\mathcal{P}$ contains the resulting relocation containers.

We use the obtained solution to tighten the lower bound on reduced costs. We update the maximum stack height $s_k^{\max}$ for $k \neq i$. Like before we sum over variables b_{ijnt}. But now, we exclude container t and containers $n \in \mathcal{P}$ from the sum. These containers are not included since they are supposed to be in stack i. We then update γ_n^{in} for pick-up containers with the updated stack height. Only if Constraint (9.9) holds, the subproblem may generate columns with negative reduced costs that pick up containers determined by the flow problem. This validity of the bound may be proved with the same mechanism as before.

$$r' + \sum_{n \in \mathcal{P}} \sum_{j=j'+1}^{j'+r} a_{nj} \cdot \gamma_{i'jnt'} - \sum_{n \in \mathcal{P}} \gamma_n^{in} + \delta_{i'j't'} - \mu_{t'} < 0 \tag{9.9}$$

For our example, only the maximum stack height of stack 3 changes: it is reduced from 1 to 0. This implies that containers can no longer be relocated to position $(3, 4)$. However, this has no impact on any γ_n^{in} since $\gamma_{ijnt'}$ for position $(3, 4)$ is not maximal for any container. Constraint (9.9) holds since $3 + (-0.61 - 0.63 - 0.35) - (0.39 + 1.00 + 0.65) = -0.63$.

Step 2 - Input of relocation containers

If Constraint (9.9) holds, we determine where to put relocation containers. Values of dual variables and the relocation rules from Heuristic HC (see Section 6.3) guide this decision. The objective is to relocate each container $n \in \mathcal{P}$ into position (k, l) for which $\gamma_{klnt'}$ is maximal. If several positions have identical $\gamma_{klnt'}$, relocation rules from Heuristic HC are used to determine the target stack k. If several possible heights l in stack k have the same benefit, one is chosen arbitrarily.

For our example, we have to determine where to relocated containers 12, 13 and 15. We start by relocating container 12. Dual variables (see Figure 9.4) do not indicate where to relocate container 12 since several $\gamma_{kl12t'}$ are maximal. According to the relocation rules of Heuristic HC, container 12 should be relocated to stack 3. The height is chosen arbitrarily, e.g. 3. In this case, container 12 is relocated to position $(3, 3)$. Then, container 13 is relocated. Dual variables indicate that it should be relocated to position $(2, 3)$. Finally, container 15 is relocated. Dual variables suggest to relocate it to positions $(3, 1)$ or $(3, 2)$. But, if container 15 is relocated to stack 3 it has to be relocated to position $(3, 4)$ above container 12. Instead, we relocate container 15 to position $(1, 2)$ with the highest attainable γ_{klnt}. The created column has a reduced cost of $3+(-0.61-0.39)+(-0.63-1.00)+(-0.35-0.60)+0.33-0.33=-0.58$.

As seen in the example, executed relocations impact relocations of subsequent containers and might block beneficial relocations. If the generated column does not have negative reduced costs, we execute step 2 again, but relocate containers from the lowest to the topmost relocation container. This increases the number of generated columns and decreases the risk that we omit beneficial columns.

After generating columns for all periods, all retrieval positions and all numbers of relocations, we add columns with negative reduced to the restricted master problem. These columns are also used to update attainable positions.

The columns generated by the heuristic subproblem depend on attainable positions and on values of variables b_{ijnt} in the restricted master problem. This prevents the subproblem from generating lots of columns that can never be part of an integer solution. But, this also implies that some columns with negative reduced costs may not be generated. However, dual variables should guide the heuristic subproblem towards promising columns that lead to relevant attainable positions and a good solution of the restricted master problem.

9.2 Heuristic branch and price

There is no guarantee that the solution of the restricted master problem is optimal since the heuristic subproblem does not necessarily generate all columns with negative reduced costs. Consequently, the obtained solution cannot be used as a lower bound on the initial integer problem. That is why, we aim to obtain good integer solutions quickly rather than solving the restricted master problem to optimum.

Section 6.3 showed that Heuristic HC performs well, even if it does not obtain optimal solutions. This implies that the used relocation rules perform well for most situations, but not for all. The restricted master problem is initialized with the solution of Heuristic HC. If the subproblem generates a new column with negative reduced costs for period t, this indicates that the overall solution might be improved by relocating containers differently at period t. We evaluate the quality of the suggested column with Heuristic HC. We apply columns obtained by Heuristic HC for periods 1 to $t-1$ and the new column for period t to the initial layout. This leads to a new layout with fewer containers. We apply Heuristic HC to the new layout.

The benefits are twofold. First, we obtain a new integer solution. Second, we generate new columns for periods $t+1$ to $T-1$ that can be added to the restricted master problem.

These new columns for $t' > t$ ensure that the new column at period t can enter the solution of the restricted master problem immediately. Without columns for periods $t' > t$ it might happen that the new column would violate consistency constraints of the bay (e.g. if it relocates container n to position (i, j), but no column in the model retrieves container n from position (i, j) in subsequent periods). In addition, new columns for $t' > t$ are used to update attainable positions and enlarge the pool of columns that can be generated in the sequel.

Heuristic HC can only be applied to integer layouts where each container is located at exactly one position (all variables b_{ijnt} take binary values). It cannot be applied to periods after the first fractional layout (some variables b_{ijnt} take fractional values) and is run only for few columns. We introduce a branching procedure to be able to run the heuristic for more columns at different periods. We branch only on integer layouts. When branching at period t, all created nodes impose the first $t - 1$ columns currently in the solution and different columns for period t. This postpones the first fractional period and makes sure that Heuristic HC can be run for later periods.

To evaluate the quality of a node, we determine lower and upper bounds on the number of relocations. We apply the columns imposed by the node to the initial layout and count the number of executed relocations. This leads to a new integer layout with fewer containers. We compute lower and upper bounds for this new layout with the procedure explained in Section 6.3 and with Heuristic HC. To obtain the lower and upper bounds of the node, we add the number of already executed relocations to the lower and upper bound of the new layout. We implemented three strategies to explore the branching tree: smallest lower bound first, smallest upper bound first and smallest sum of lower and upper bound first. To avoid memory problems we reduce the size of the branching tree by omitting nodes with an upper bound above a given threshold.

The solution time of the restricted master problem increases with the number of added columns. To speed up its solution, we remove columns from the model if a threshold is overpassed. In this case, we remove all columns with a reduced cost above a given limit from the model. Only for periods with few columns, no columns are removed.

Algorithm 9.2 describes the complete solution procedure. The restricted master problem and attainable positions are initialized. Every node is solved using column generation with the heuristic subproblem. If possible, we run Heuristic HC for columns added to the model. These columns are added to the restricted master problem and used to update attainable positions. If the number of columns in the restricted master problem reaches the threshold, columns are removed based on their reduced costs. If no more columns are generated for the node, we branch and create new nodes. Columns generated during the branching procedue are also added to the restricted master problem and used to update attainable positions. The solution procedure ends if no more nodes have to be handled and the best integer solution is returned.

9.3 Computational results

This section evaluates different variants of the heuristic branch and price approach to determine which parameter settings obtain the best results. It also compares the obtained results to other approaches from literature.

Algorithm 9.2 Heuristic branch and price approach

```
if optimality criterion holds then
  return optimal heuristic solution
end if
initialize attainable positions
rootNode ← initialize restricted master problem (RMP)
nodesToHandle ← rootNode
repeat
  node ← get node from nodesToHandle
  repeat
    solve node with column generation with heuristic subproblem
    run heuristic HC
    add generated columns and update attainable positions
    remove columns
  until no new columns were generated
  create new nodes
  nodesToHandle ← new nodes
  add generated columns and update attainable positions
until nodesToHandle is empty
return best integer solution
```

For the heuristic branch and price approach, we have to decide how much time should be spent to solve each node and which columns should be kept in the master problem. This is no easy choice. If each node is solved to optimum and all columns are kept in the master problem, chances are higher to create those columns that may lead to a better solution. But, only few nodes can be explored. If we limit the time spent to solve each node and keep few columns in the master problem, more nodes may be explored. But, promising columns may not be generated. Several parameters define the columns to be added and removed, the branching strategy and time limits:

- *maxAdd*: maximum reduced cost for which a column is added to the master problem;
- *maxUpdate*: maximum reduced cost for which a column is used to update attainable container positions;
- *maxNbCol*: threshold above which columns are removed from the master problem;
- *minNbColPeriod*: minimum number of columns per period to remove columns;
- *maxRem*: maximum reduced cost above which a column is removed;
- *brStrat*: chosen branching strategy;
- *brOrder*: order in which nodes are handled;
- *maxUB*: threshold above which a new node is omitted;
- *timeGlobal*: global time limit;
- *timeNode*: time limit per node.

Table 9.1: Experimental settings to evaluate different variants of the heuristic branch and price approach

General parameters:	
$maxNbCol = 50 \cdot N$	$minNbColPeriod = 10$
$maxUB = 1.25 \cdot$ best integer solution	$timeLimit = 5$min

Variant	*maxAdd*	*maxUpdate*	*maxRem*	*brStrat*	*brOrder*	*timeNode*
BP-A	0-	0+	0+	-	-	-
BP-B	0-	0+	0+	last int	LB + UB	-
BP-C	0-	0+	0+	last int	LB	-
BP-D	0-	0+	0+	last int	UB	-
BP-E	0-	0+	0+	t_{node}+1	LB + UB	-
BP-F	0-	0+	0+	t_{node}+1	LB	-
BP-G	0-	0+	0+	t_{node}+1	UB	-
BP-H	0-	0+	0+	t_{node}+1	LB + UB	$1.0 \cdot N$
BP-I	0-	0+	1+	t_{node}+1	LB + UB	-

Table 9.1 summarizes the different analyzed variants. Parameters *maxNbCol*, *minNbColPeriod*, *maxUB* and *timeGlobal* are identical for all variants and only displayed once. Variant *BP-A* adds columns with reduced costs < 0 to the restricted master problem and uses columns with reduced costs ≤ 0 to update possible container positions. When removing columns, all columns with reduced costs > 0 are removed. Variant *BP-A* does not branch but applies the Heuristic *HC* as often as possible for new columns. Variants *BP-B* to *BP-G* test different branching strategies. Variants *BP-B* to *BP-D* branch on the last integer layout in the solution. Variants *BP-E* to *BP-G* branch on the first period for which the current node does not impose a column. In both cases, a node is generated for all columns in the restricted master problem for the given period (also for those not in the solution). Variants *BP-B* and *BP-E* handle the node with the smallest LB + UB first, variants *BP-C* and *BP-E* the node with the smallest LB and variants *BP-D* and *BP-G* the node with the smallest UB. Variant *BP-H* limits the time that is spent on each node. Variant *BP-I* increases the threshold above which columns are removed from the model.

Experiments are carried out on a computer with Intel(R) Xeon(R) CPU clocked at 2.67GHz (dual core), 3.48GB RAM and operating with Windows XP Professional. We use Cplex 12.1 to solve the master problem and the network flow problem in the subproblem. Experiments are run on a sample of 10 instances for different bay sizes. The time limit is set to 5 minutes. If the time limit is reached or the solution process stops due to memory limits, the best integer solution obtained so far is returned.

Tables 9.2 and 9.3 compare the solution quality of variants *BP-A* to *BP-I*. Table 9.2 reports the best integer solution obtained with each variant for each instance. Optimal solutions are marked in bold; solutions identical to the solution on Heuristic *HC* in italic. Table 9.3 provides more details on the different variants. It reports average values for the number of relocations and indicates the number of instances solved to optimum. It also shows the number of handled and remaining nodes, the number of times the subproblem and the Heuristic *HC* were executed and the number of columns added and removed from

Table 9.2: Number of relocations for variants *BP-A* to *BP-I* for sample instances

Instance	*BP-A*	*BP-B*	*BP-C*	*BP-D*	*BP-E*	*BP-F*	*BP-G*	*BP-H*	*BP-I*
4-5-8	17	16	17	16	17	17	16	17	17
4-6-27	16	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**
4-7-35	21	**20**	21	**20**	**20**	21	**20**	**20**	**20**
5-4-24	18	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**
5-5-31	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**	**24**
5-6-32	29	**24**	25	**24**	**24**	25	**24**	26	**24**
5-7-30	29	29	29	29	29	29	29	29	29
6-6-29	39	39	39	39	39	39	39	39	39
6-10-32	63	63	63	63	63	63	63	63	63
10-10-1	*144*	*144*	*144*	*144*	*144*	*144*	*144*	*144*	*144*

the master problem. It also presents the percentage of time spent in the master problem, in the subproblem and creating nodes during branching.

Variant *BP-A* solves the problem at the root node. It improves the heuristic solution for some instances, but obtains the optimal solution only once. As discussed above, the heuristic can be applied only to integer layouts. Here, it is run only few times and few solutions are explored.

Variants *BP-B* to *BP-D* obtain better results. They branch on the last integer layout of the master problem. This increases the number of explored nodes and consequently the number of added columns and the number of times the Heuristic *HC* is run. They obtain optimal solutions for several instances. But, for instance 10-10-1 the heuristic solution is not improved. For this instance, the master problem is long to solve. Consequently, the subproblem and the heuristic are called only few times and the branching procedure is not started.

Variants *BP-E* to *BP-G* branch on each period. But, they mostly evaluate the same nodes as *BP-B* to *BP-D* within the given time limit. Consequently, the performance of *BP-E* to *BP-G* is similar to the performance of *BP-B* to *BP-D*.

Comparing the order in which nodes are handled shows that UB and LB + UB perform slightly better than LB. This shows that the lower bound provides a poor estimation on the number of relocations and misjudges nodes.

Variant *BP-H* imposes a time limit on each node to avoid that too much time is spent on each node. The number of explored nodes increase, but the solution quality is not increased. The subproblem and the heuristic are called fewer times, since the time limit on a node may prevent their execution. If the time limit is reached while solving the master problem, the master problem is not aborted. Thus, more time than imposed by the time limit may be spent on a node.

For instances *BP-B* to *BP-I*, around 98% of all added columns are removed. Removed columns may be added to the master problem later on. Adding columns previously removed from the master problem, accounts for around 22 to 24% of added columns. Variant *BP-I*

Table 9.3: Performance comparison of variants *BP-A* to *BP-I*

Variant	Best sol [nb rel]	Inst opt	Nodes handled	Nodes remain	Calls sub	Calls heur
BP-A	40.0	1	1.0	0.0	48.4	32.3
BP-B	39.0	5	13.0	17.4	599.3	104.1
BP-C	39.3	3	13.0	25.4	609.3	107.7
BP-D	39.3	5	17.6	26.3	689.1	120.3
BP-E	39.1	5	16.7	18.6	2 291.1	1 758.8
BP-F	39.3	3	17.6	26.3	689.1	120.3
BP-G	39.0	5	15.0	16.1	577.6	106.3
BP-H	39.3	4	21.0	25.6	706.6	210.0
BP-I	39.1	5	18.6	19.8	574.0	94.2

Variant	Columns added	Columns removed	Time mas [%]	Time sub [%]	Time bra [%]
BP-A	9 250.2	7 939.9	88.3	10.6	0.0
BP-B	84 485.5	83 067.8	69.1	30.1	0.0
BP-C	80 848.3	79 435.2	70.7	28.5	0.0
BP-D	84 714.7	83 234.0	70.4	28.8	0.0
BP-E	87 077.5	85 597.0	62.5	36.7	0.0
BP-F	84 714.7	83 234.0	70.4	28.8	0.0
BP-G	84 125.6	82 630.6	71.0	28.2	0.0
BP-H	93 825.2	92 247.3	70.5	28.6	0.0
BP-I	45 215.1	42 218.9	76.0	23.0	0.0

shows the impact of removing fewer columns from the model. It removes around 93% of added columns, but adding previously removed columns still accounts for around 17% of added columns. *BP-I* performs worse than other variants. More time is spent on solving the master problem, because of the increased number of variables. This reduces the number of explored nodes and the number of times the subproblem and the heuristic are called.

On average more than 70% of the solution time is spent in the master problem and less than 36% in the subproblem. The time spent in the master problem increases with the size of the instance. It exceeds 95% for instances 6-10-32 and 10-10-1. As mentioned before, this decreases the overall performance since the subproblem and the heuristic are executed only few times. The time needed for creating nodes and branching can be neglected.

We run *BP-H* on all instances with a time limit of 5 minutes. Table 9.4 compares *BP-H* to existing solution approaches relocating only blocking containers. It indicates the average number of relocations and the average solution time per instance set for each solution approach. *KH* refers to the heuristic proposed by Kim and Hong (2006), *HC* to the slightly modified heuristic of Caserta et al. (2012), *CM* to the corridor method from Caserta, Voß and Sniedovich (2011), *RB* to the random based procedure from Caserta et al. (2009), *RCM* to the random based procedure using the corridor method from Caserta and Voß (2009) and IDA^* to the approach from Zhu et al. (2012). For *CM*, a feasible solution is not obtained for

all instances and the average is computed only with solved instances (see Zhu et al. (2012)). For RCM, results are only published for the biggest instances.

Variant $BP\text{-}H$ obtains better results than KH, HC, CM and RB, but worse results than RCM and IDA^*. For instances 3-3 to 4-5, the time needed to find the best integer solution is comparable with run times of other approaches. For bigger instances, the time needed to find the best integer solution increases and overpasses solution times of other approaches.

Only Zhu et al. (2012) publish results for single instances[1]. Table 9.5 compares their results and our results for non-trivial instances in more detail. It shows the average number of relocations for IDA^* and $BP\text{-}H$ for non-trivial instances and counts the number of instances where one approach outperforms the other. It also compares the number of evaluated relocations. For IDA^*, each node in the branching tree corresponds to exactly one relocation. For $BP\text{-}H$, we know the number of different columns added to the master problem. But, each column may contain several relocations. To obtain an upper bound on the number of explored relocations we multiply the number of columns with the maximum number of relocations per column $(H - 1)$.

Results show that IDA^* obtains better results than $BP\text{-}H$ for several instances of different instance sets. $BP\text{-}H$ obtains better results than IDA^* on 12 instances for instance sets 5-9, 5-10 and 6-10. The exact values of explored relocations cannot be compared, but we may use them to extract some tendencies. For instance sets 3-3 to 5-5, IDA^* evaluates fewer relocations then $BP\text{-}H$. When exploring the branching tree, it updates the global lower bound and stops if the best found solution equals the global lower bound. $BP\text{-}H$ does not update the global lower bound and stops if all nodes are explored or if the time limit is reached. For instance sets 5-6 to 10-10, IDA^* explores much more nodes, but obtains only slightly better results. This shows that the column generation approach guides the solution into the right direction. But, results also show that the overall heuristic branch and price approach has to be improved to obtain better results.

[1]Results can be downloaded from `\http://www.zhuwb.com/crp`

Table 9.4: Average number of relocations for different solution approaches for CRP relocating only blocking containers

	KH		*HC*		*CM*		*RB*		*RCM*		*IDA**		*BP-H*	
PC used	Pentium IV 512 MB RAM		Intel(R) Xeon 2.67 GHz 3.48 GB RAM		Pentium IV 512 MB RAM		Core 2 Duo 2 GHz 2 GB RAM		Pentium IV 512 MB RAM		Intel Core i7 2.66 GHz 12 GB RAM		Intel(R) Xeon 2.67 GHz 6.48 GB RAM	
Inst	nbRel	time[s]	nbRel	time[s]	nbRel	time[s]	nbRel	time[s]	nbRel	time[s]	nbRel	time[s]	nbRel	best int [s]
3-3	7.1	0.1	5.1	0.1	5.4	0.1	5.1	60.0			5.0	1.0	5.0	0.1
3-4	10.7	0.1	6.3	0.1	6.5	0.1	6.2	60.0			6.2	1.0	6.2	0.1
3-5	14.5	0.1	7.1	0.1	7.3	0.1	7.1	60.0			7.0	1.0	7.0	0.1
3-6	18.1	0.1	8.5	0.1	7.9	0.2	8.6	60.0			8.4	1.0	8.4	0.2
3-7	20.1	0.1	9.3	0.1	8.6	0.1	9.4	60.0			9.3	1.0	9.3	0.1
3-8	26.0	0.1	10.7	0.1	10.5	0.2	10.8	60.0			10.7	1.0	10.7	0.1
4-4	16.0	0.1	11.0	0.1	9.9	0.1	10.3	60.0			10.2	1.0	10.4	0.2
4-5	23.4	0.1	13.6	0.1	16.5	0.5	13.4	60.0			13.0	1.0	13.1	0.4
4-6	26.2	0.1	14.7	0.1	19.8	0.5	14.4	60.0			14.0	1.0	14.1	2.0
4-7	32.2	0.1	16.9	0.1	21.5	0.5	16.6	60.0			16.1	1.0	16.2	4.5
5-4	23.7	0.1	16.8	0.1	16.6	0.5	15.7	60.0			15.4	1.0	15.6	15.4
5-5	37.5	0.1	21.2	0.1	18.8	0.5	19.8	60.0			18.9	1.0	19.2	27.2
5-6	45.5	0.1	24.3	0.1	22.1	0.8	23.1	60.0			22.1	1.0	22.5	28.6
5-7	52.3	0.1	26.3	0.1	25.8	0.8	24.9	60.0			24.3	1.0	24.7	41.0
5-8	61.8	0.1	29.6	0.1	30.1	0.8	29.1	60.0			27.9	1.0	28.2	21.2
5-9	72.4	0.1	32.4	0.1	33.1	1.4	32.0	60.0			30.7	1.0	30.9	41.1
5-10	80.9	0.1	35.5	0.1	36.4	1.9	34.8	60.0			33.6	1.0	33.8	56.3
6-6	37.3	0.1	35.9	0.1	32.4	1.7	32.6	60.0	30.9	60.0	31.1	1.0	33.0	44.6
6-10	75.1	0.1	49.9	0.1	49.5	2.0	47.8	60.0	46.2	60.0	47.2	1.0	48.0	66.0
10-6	141.6	0.1	101.3	0.1	102.0	4.7	83.6	60.0	76.6	60.0	85.0	1.0	97.6	32.3
10-10	178.6	0.2	139.3	0.1	128.3	6.3	121.3	60.0	105.5	60.0	126.3	1.0	136.7	13.6
Average	47.7	0.1	29.3	0.1	29.0	1.1	27.0	60.0	-	-	26.8	1.0	28.1	18.8

Table 9.5: Comparison of IDA^* and $BP\text{-}H$ for non-trivial instances

Inst	IDA^* nbRel	$BP\text{-}H$ nbRel	Nb. $IDA*$ better	Nb. $BP\text{-}H$ better	$IDA*$ rel	$BP\text{-}H$ columns	$BP\text{-}H$ max rel
3-3	6.8	6.8	0	0	17.7	36.0	144.0
3-4	7.5	7.6	1	0	47.0	109.0	436.0
3-5	8.8	8.8	0	0	60.6	89.3	357.0
3-6	11.5	11.5	0	0	889.3	2 094.5	8 378.0
3-7	11.7	11.7	0	0	848.7	284.9	1 139.4
3-8	12.3	12.3	0	0	424.1	543.9	2 175.5
4-4	10.7	10.9	6	0	218.0	2 415.2	12 075.9
4-5	13.6	13.7	3	0	2 015.2	21 672.6	108 362.8
4-6	14.6	14.6	2	0	16 967.5	21 941.3	109 706.6
4-7	16.9	16.9	1	0	14 831.7	26 077.4	130 386.8
5-4	15.6	15.8	6	0	5 021.9	44 523.6	267 141.3
5-5	18.8	19.1	11	0	87 344.2	73 642.1	441 852.3
5-6	22.3	22.7	10	0	1 160 108.1	93 442.2	560 653.3
5-7	24.8	25.3	12	0	2 850 340.3	73 754.1	442 524.9
5-8	28.1	28.4	8	0	4 489 678.1	64 369.3	386 215.5
5-9	30.8	31.1	10	3	5 430 310.5	49 873.5	299 240.8
5-10	33.9	34.1	8	4	5 224 125.4	34 998.9	209 993.5
6-6	31.1	33.0	27	0	7 162 773.0	82 292.4	576 047.0
6-10	47.5	48.3	15	5	9 070 816.0	13 004.2	91 029.1
10-6	85.0	97.6	40	0	12 522 798.9	12 753.4	140 287.7
10-10	126.3	136.7	38	0	9 404 815.6	7 174.0	78 913.5
Average	27.5	28.9	198.0	12.0	2 735 450.1	29 766.3	184 145.7

9.4 Conclusion

This chapter presented a heuristic branch and price approach for the container relocation problem. Its objective is to obtain good integer solutions rather than the optimal linear relaxation of the initial problem. A heuristic subproblem generates columns to be added to the restricted master problem. It works in two steps. First, a network flow problem determines relocation containers to be picked. To do so, it uses values of dual variables, the current primal solution of the master problem and attainable positions. It then determines to which positions containers should be relocated to. To do so, it uses values of dual variables and heuristic relocation rules.

To obtain new integer solutions, the Heuristic *HC* is run for columns generated by the heuristic subproblem. The heuristic can only be run on integer layouts. To explore columns generated after the first fractional layout, the heuristic column generation approach is embedded in a branching procedure. We compared different variants of this heuristic branch and price approach to evaluate different branching strategies and the impact of different input parameters. One variant was then compared to existing solution approaches. It outperformed some approaches and obtained some new best known solutions. However, further improvements are necessary to become more competitive.

Several aspects of the branch and price approach can potentially be improved. Improving the quality of lower and upper bounds should improve the obtained results. The heuristic branch and price, uses lower and upper bounds presented in Section 6.3. Instead, it could be implemented with the improved version of these bounds presented in the same section. In addition, more experiments could be run to adjust input parameters.

Results show that most of the time is spent solving the master problem. Consequently, the subproblem and the heuristic are called few times and few solutions are explored. Probably, the solution approach would obtain better results if the solution time of the master problem can be decreased. This is especially true for bigger instances.

Chapter 10

Dynamic container relocation problem

The academic container relocation problem assumes that the entire retrieval sequence is known in advance. This is realistic for vessels where the stowage plan is known ahead in time. But, exact truck arrivals can hardly be forecasted and are revealed over time. Consequently, the retrieval order is not known in advance. This chapter deals with a dynamic and more realistic version of the container relocation problem, where information about container retrievals becomes revealed over time.

Section 10.1 introduces the dynamic container relocation problem and presents related literature. Section 10.2 discusses the expected value of relocations as a criteria to evaluate the quality of a given bay layout. Section 10.3 details different relocation strategies. Section 10.4 evaluates the performance of these strategies. Section 10.5 concludes the chapter.

10.1 Introduction

10.1.1 Problem description

Container terminals have limited information on exact arrival times and on the arrival sequence of trucks. It is not uncommon, that terminals obtain this information only when trucks check in at the terminal gate. When processing the truck at the terminal gate, a container request is issued to retrieve the corresponding container from the storage area. The terminal operator has to decide in which order to serve the current container requests and where to relocate blocking containers. The decision is based on known requests, since the terminal operators has no information on future retrievals. Figure 10.1 illustrates the dynamic container relocation problem. The objective is to minimize truck service times.

The order in which requests are served impacts truck service times and the number of relocations. Our main objective is to evaluate the benefit of knowing the retrieval sequence ahead in time, rather than evaluating different service policies. We suppose that trucks are served with a first-come, first-served policy. In this case, truck service times depend mainly on the number of relocations. Our objective is to minimize the number of relocations. Assumptions A2 to A8 from the container relocation problem remain valid for the dynamic container relocation problem. Assumption A1 (known retrieval order) is replaced by assumption A9 and assumption A10 is added.

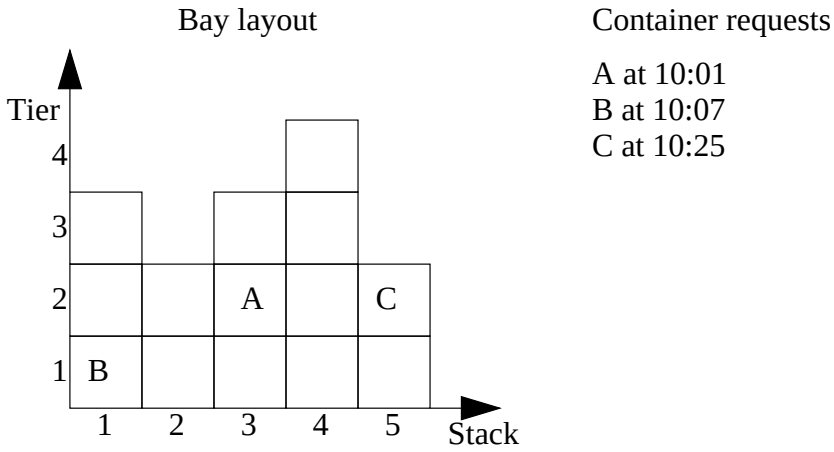


Figure 10.1: Dynamic container relocation problem

A2: No new containers arrive during the retrieval process.

A3: Only the topmost container of a stack can be picked up. A relocated container can only be put on the top of another stack or on the ground.

A4: Containers are only relocated within the bay since relocations between bays are very time consuming.

A5: The bay size is limited by the maximum numbers of stacks and tiers.

A6: Containers in the same bay have the same size and can be piled up in any order.

A7: The distance traveled within one bay (horizontally and vertically) has little impact on the time to relocate or to retrieve containers.

A8: Only blocking containers located above the current target container may be relocated.

A9: Container requests become known when trucks are processed at the terminal gate.

A10: Trucks are served with a first-come, first-served policy.

We recall the notation introduced in Chapter 6 for the container relocation problem. We use the same notation for the dynamic container relocation problem. A bay consists of W stacks and H tiers. Each slot within the bay is addressed with coordinates (i, j) where $i \in \{1, \ldots, W\}$ and $j \in \{1, \ldots, H\}$. The initial configuration contains N containers, labeled $1, \ldots, N$. Containers have to be retrieved in ascending order, e.g. container 1 is the first one to be retrieved and container N the last one. At each time period t ($t = 1, \ldots, T$), container $n = t$ has to be retrieved and several containers may be relocated. Contrary to previous chapters, container labels are not known from the beginning, but revealed over time. To represent partly knowledge about the future retrieval sequences, we introduce a look-ahead horizon D ($D \geq 1$). It indicates that at each period t the exact retrieval sequence for the next D containers is known: at period t, $D_t^a = t$ is the first known retrieval container and $D_t^b = t + D - 1$ the last known retrieval container.

10.1.2 Related literature

To the best of our knowledge no scientific literature exists on the dynamic container relocation problem. But, several articles deal with the related stacking problem. The aim is to find good storage positions for incoming containers based on partial knowledge about their destinations, weights and departure times. The main objectives are to use the storage space efficiently, to reduce traveling times within the terminal and to reduce the number of relocations. Here, we only present studies aiming to minimize the number of relocations.

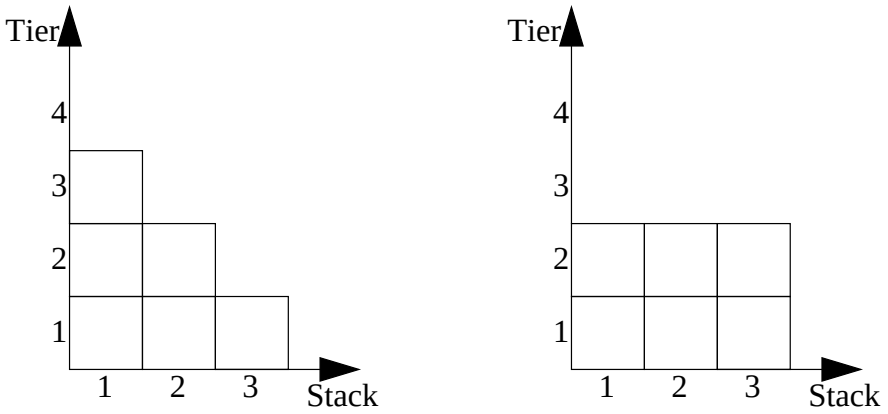
Dekker et al. (2006) and Borgman et al. (2010) evaluate different stacking strategies and the impact of available information via simulation. The performance of each strategy is measured via the number of relocations, the yard crane workload and the level of occupancy of the yard. They show that stacking containers on ground positions reduces the number of relocations. They compare scenarios with no information on future retrievals with scenarios with imprecise information on future departure times. Results show that using imprecise information increases the efficiency of the terminal. Park et al. (2011) present an online search algorithm to decide where to stack incoming containers. The algorithm tries variants of the best-so-far policy and can easily adapt to changes at the terminal. Results show that this algorithm can reduce quay crane delays, but does not obtain the best results for average truck waiting times.

Zhao and Goodchild (2010) use simulation to evaluate the use of information on truck arrivals to reduce relocations during the retrieval process. They run experiments for different levels of information and different bay configurations. Results show that already limited information on future arrivals can reduce the number of relocations. They also show that updating information in real time lowers information requirements. Jang et al. (2013) consider the problem with groups of homogeneous containers. They present a genetic algorithm for the case where the retrieval order of groups is known. They also present a statistical model to estimate the expected number of relocations when no information on future retrievals is available.

Yang and Kim (2006) consider the problem of stacking incoming containers in a way that minimizes the expected number of relocations. They relocate each container at most once. They address a static and a dynamic version of the problem. For the static problem, arrival and due dates of all containers are known in advance; for the dynamic version, arrival and due dates become known when containers arrive at the terminal. They use dynamic programming and a genetic algorithm to solve the static problem and use heuristics based on known departure times to solve the dynamic problem. Preston and Kozan (2001) present a container location model that minimizes the time needed to transfer containers from the storage area to vessels. This model includes traveling and relocation times. They formulate a mixed integer linear programming model and solve the problem via a genetic algorithm.

10.2 Expected value of relocations

This section introduces a criteria to indicate the quality of a given bay layout if we do not have any information on future retrievals. In this case, all containers in the bay are equally likely to be retrieved next. We determine the expected value of relocations, EVR, necessary to retrieve one container from the given bay. To do so, we compute the average number



(a) Layout A, $EVR = 0.67$

(b) Layout B with minimal EVR, $EVR = 0.5$

Figure 10.2: Expected value of relocations EVR for two different layouts with 6 containers

of blocking containers. Equation (10.1) defines the expected value EVR. It depends on the number of containers in the bay, N', and on the number of containers per stack, $s(i)$ for all $i = 1, \ldots, W$. Figure 10.2 illustrates the computation on two examples: $EVR_A = 1/6 \cdot ((0+1+2)+(0+1)+(0)) = 0.67$ and $EVR_B = 1/6((0+1)+(0+1)+(0+1)) = 0.5$.

$$EVR = \frac{1}{N'} \cdot \sum_{i=1}^{W} \sum_{j=0}^{s(i)-1} j \tag{10.1}$$

Lemma 1. *The minimum difference between the lowest stack and the highest stack equals 0 if* $N' \mod W = 0$ *and 1 if* $N' \mod W \neq 0$.

Proof. The minimum difference between the lowest stack and the highest stack is obtained if containers are evenly distributed among stacks. If $N' \mod W = 0$, each stack has a height of N'/W; if $N' \mod W \neq 0$ some stacks have height $\lceil N'/W \rceil$ and others $\lfloor N'/W \rfloor$. □

Lemma 2. *The expected value of relocations EVR is minimal if the difference between the lowest stack and the highest stack is minimal.*

Proof. We assume that for layout 1 the difference between the lowest stack and the highest stack is not minimal. Let a be the lowest stack in layout 1 and b the highest stack. We obtain layout 2 by moving the topmost container from stack b to stack a. Let EVR_{ab} be the expected value of layouts 1 and 2 without stacks a and b. We compare the expected value of layouts 1 and 2.

$$\begin{aligned} EVR_2 - EVR_1 &= \left(EVR_{ab} + \sum_{j=0}^{s(a)} j + \sum_{j=0}^{s(b)-2} j \right) - \left(EVR_{ab} + \sum_{j=0}^{s(a)-1} j + \sum_{j=0}^{s(b)-1} j \right) \\ &= s(a) - s(b-1) \end{aligned}$$

It is hence possible to reduce EVR by moving one container from the highest stack b to the lowest stack a as long as $s(a) < s(b) - 1$. Consequently, EVR is minimal if the difference between stacks a and b is minimal. □

10.3 Different relocation strategies

This section presents different relocation strategies that may be applied to the dynamic container relocation problem for a partial known retrieval order of length D.

Strategy S1: Random heuristic

For each container to be relocated, the heuristic randomly chooses a stack that is not full.

Strategy S2: Leveling heuristics for $D = 1$ and $D = 2$

The objective of the leveling heuristic is to relocate containers in a way that minimizes the expected value of relocations EVR. Lemma 1 and 2 show that containers should be distributed equally over stacks to minimize EVR.

For $D = 1$, only the current retrieval container is known. For each container to be relocated, the heuristic determines current stack heights and relocates the container to the lowest stack. If several stacks have the same height, the leftmost stack among them is chosen. For $D = 2$, the heuristic uses information about the second retrieval container to keep it accessible. Like before it balances stack heights by relocating containers to the lowest stack. But containers are only relocated on top of the second retrieval container if no other positions are free. If the second retrieval container itself has to be relocated, it is relocated to the highest stack.

The subsequent strategies S3 to S8 determine a partial solution to retrieve the next D containers with a minimum number of relocations. We use an adapted version of the model presented in Section 7.2.1 to do so (details are given below).

Strategy S3: Relocations are updated every time new information becomes available

The problem is solved repeatedly for each period $t = 1, \ldots, T$ with information on containers $n = D_t^a, \ldots, D_t^b$. We initialize the model for period $t = 1$ with variables and constraints corresponding to periods $t = D_1^a, \ldots, D_1^b$ and solve it. We then adapt the model to the next period $t = t + 1$ by adding variables and constraints corresponding to period D_t^b (since variables and constraints corresponding to periods D_t^a to $D_t^b - 1$ are already in the model). Since it is not possible to revoke decisions taken at earlier periods we fix variables representing container positions at the beginning of period t according to the solution obtained in the previous iteration. We solve the updated model. The process ends when the time horizon is reached. At each iteration, the objective function (10.2) minimizes the number of relocations necessary to retrieve all D containers for the given initial layout.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} \tag{10.2}$$

Strategy S4: Relocations are determined for the next D containers and are not updated

The complete relocation sequence to retrieve the next D containers is determined with the information on these D containers. The solution is not updated if new information becomes available. The problem is solved repeatedly at periods 1, $1+D$, $1+2D$, ... with information on the next D retrieval containers. We initialize the model for period $t = 1$ with variables and constraints corresponding to periods $t = D_1^a, \ldots, D_1^b$ and solve it. We then adapt the model to the next iteration at period $t = t + D$ by adding variables and constraints for periods D_t^a to D_t^b. To prevent revoking decisions taken at earlier periods, we set variables representing container positions at the beginning of periods $t - D + 1$ to t to the values obtained in the previous iteration. We solve the updated model. The process ends when the time horizon is reached. At each iteration, the objective function (10.3) minimizes the number of relocations necessary to retrieve all D containers for the given initial layout.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} \tag{10.3}$$

Strategy S5: No detailed information about far-away retrievals

We suppose that we know the exact retrieval sequence of the next D containers. In addition, we know the subsequent D' containers to be retrieved, but not their exact retrieval order. Keeping subsequent retrieval containers D' on top of stacks should reduce the number of relocations necessary at the next iteration to retrieve these containers. We introduce integer variables a_{ijnt} that count the number of containers located above the next retrieval containers $n \in D'$:

$$a_{ijnt} = \begin{cases} 0 & \text{if container } n \text{ is not located at position } (i,j) \text{ at the beginning of period } t, \\ R & \text{number of containers above container } n \text{ if it is located at position } (i,j) \\ & \text{at the beginning of period } t. \end{cases}$$

The objective function (10.4) penalizes the number of relocations for the current iteration (periods D_t^a to D_t^b). In addition, it penalizes the number of containers above subsequent retrieval containers at the beginning of the next iteration at period D_t^b+1. Constraint (10.5) defines variables a_{ijnt} for period $D_t^b + 1$ for the next D' retrieval containers.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} + w_1 \cdot \sum_{i=1}^{W} \sum_{j=1}^{H-1} \sum_{n \in D'} a_{ijnD_t^b+1} \tag{10.4}$$

$$\sum_{j'=j+1}^{H} \sum_{n' \in N \setminus \{n\}} b_{ij'n'D_t^b+1} \leq a_{ijnD_t^b+1} + (H-1) \cdot (1 - b_{ijnD_t^b+1}) \quad \forall i = 1, \ldots, W, j = 1, \ldots, H-1, n \in D' \tag{10.5}$$

With strategies S6 and S7, we want to analyze the impact of different intermediate bay layouts on the total number of relocations. The objective is to be able to determine layouts

that are advantageous with regard to unknown future retrievals. We compare two cases: distribute containers evenly among stacks and keep one stack empty. Again, only the initial layout for the next iteration (the layout at period $D_b^t + 1$) is of interest.

Strategy S6: Distribute containers evenly among stacks

The objective is to distribute containers evenly among all stacks to reduce the expected value EVR. We use integer variables a_{ijnt} (introduced above) to count the number of containers located above each container. The expected value EVR is identical to the total of all a_{ijnt}. The objective function (10.6) penalizes the number of relocations and to minimizes the expected value at period $D_t^b + 1$. Constraint (10.7) defines variables a_{ijnt} for period $D_t^b + 1$ for all containers.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} + w_1 \cdot \sum_{i=1}^{W} \sum_{j=1}^{H-1} \sum_{n=D_t^b+1}^{N} a_{ijnD_t^b+1} \tag{10.6}$$

$$\sum_{j'=j+1}^{H} \sum_{n' \in N \setminus \{n\}} b_{ij'n'D_t^b+1} \leq a_{ijnD_t^b+1} + (H-1) \cdot (1 - b_{ijnD_t^b+1}) \quad \forall i = 1, \ldots, W, j = 1, \ldots, H-1, n = D_t^b + 1, \ldots, N \tag{10.7}$$

Strategy S7: Keep one stack free

The objective is to obtain a layout with at least one empty stack. This increases the expected value EVR. It might nevertheless be beneficial to have an empty stack to place containers in the next iteration. We add integer variables f_{it} and e_t to determine if at least one stack is empty.

$$f_{it} = \begin{cases} 0 & \text{if stack } i \text{ is empty at the beginning of period } t, \\ 1 & \text{otherwise;} \end{cases}$$

$$e_t = \begin{cases} 1 & \text{if at least one stack } i \text{ is empty at the beginning of period } t, \\ 0 & \text{otherwise.} \end{cases}$$

The objective function (10.8) penalizes the number of relocations and rewards an empty stack at period $D^b - t + 1$. Constraint (10.9) makes sure that f_{it} equals 0 only if stack i is empty. Constraint (10.10) determines if at least one empty stack exists.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} - w_3 \cdot e_{D_t^b+1} \tag{10.8}$$

$$\sum_{j=1}^{H} \sum_{n=D_t^b+1}^{N} b_{ijnD_t^b+1} \leq H \cdot f_{i,D_t^b+1} \qquad \forall i = 1, \ldots, W \tag{10.9}$$

$$\sum_{i=1}^{W} f_{i,D_t^b+1} + e_{D_t^b+1} \leq W \tag{10.10}$$

Strategy S8: First-come, first served policy not necessary

We want to analyze the impact of being able to serve the next D trucks in any order. This should decrease the number of relocations. A container blocking a retrieval container may itself be a retrieval container. In this case, it can be retrieved directly, rather than being relocated. Until now we imposed, that container n is retrieved at period n. Now, container n may be retrieved at any period $n - D + 1, \ldots, n + D - 1$.

The objective function (10.11) minimizes the number of relocations. It takes into account that some containers $n < t$ may be relocated at period t. Constraints (10.12) imposes that each container is retrieved within its time window. Constraint (10.13) makes sure that each container is retrieved exactly once. Variables x_{ijklnt} and b_{ijnt} for $n \leq t$ have to be added to existing constraints. A part from this, constraints remain identical and are not repeated here.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'-D+2}^{N} x_{ijklnt'} \tag{10.11}$$

$$\sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{t'=t-D+1}^{t+D-1} y_{ijnt'} = 1 \qquad \forall n = 1, \ldots, N \tag{10.12}$$

$$\sum_{n=t-D+1}^{t+D-1} y_{ijnt} = 1 \qquad \forall t = 1, \ldots, T \tag{10.13}$$

10.4 Computational results

We test strategies S1 to S8 on the instances sets 3-3, 3-4, 3-5, 3-6, 3-7, 3-8, 4-4, 4-5, 4-6, 4-7, 5-4 and 5-5 introduced by Caserta et al. (2012) and hence on $12 \cdot 40 = 480$ instances. All experiments are carried out on a computer with Inter(R) Xeon(R) CPU clocked at 2.67GHz (dual core), 3.48GB RAM and operating with Windows XP Professional. We limit the run time to 60 minutes per instance. Cplex 12.1 is used to solve the mixed integer programming models for S3 to S8.

Table 10.1 summarizes the experimental settings. We test these strategies with different look-ahead horizons $D = 1$, 2, 3, 5 and 7. For S5, we set $D' = D$. We impose a strict hierarchy to i) minimize the number of relocations per iteration and ii) optimize the initial bay layout for the next iteration. Weights w_1, w_2 and w_3 are defined based on the following observations. A relocation has a cost of 1. For S5, the maximum layout cost is obtained if all D' containers are located at height 1 and $H - 1$ containers are located above. For S6, the maximum layout cost is obtained if containers are stacked as high as possible. In this case, $\lfloor \frac{N}{H} \rfloor$ stacks contain H containers and one stack contains $N - H \cdot \lfloor \frac{N}{H} \rfloor$ containers. For S7, the maximum benefit from one empty stack should be lower than the cost of one relocation.

Table 10.2 presents experimental results. It displays the numbers of solved instances (out of 480), the average numbers of relocations and the average run times of service strategies

Table 10.1: Experimental setting for evaluating relocation strategies

Strategy	Look-ahead horizon D	Weights w
S1	n.a.	
S2	1, 2	
S3	3, 5, 7	
S4	3, 5, 7	
S5	3, 5, 7	$w_1 = (D' \cdot (H-1)+1)^{-1}$
S6	3, 5, 7	$w_2 = (\lfloor \frac{N}{H} \rfloor \cdot \sum_{i=0}^{H-1} i + \sum_{i=0}^{N-H\cdot\lfloor \frac{N}{H} \rfloor -1} i + 1)^{-1}$
S7	3, 5, 7	$w_3 = 0.5$
S8	3	

Table 10.2: Performance of relocation strategies S1 to S8 for different look-ahead horizons

Strategy	S1	S2-1	S2-2
Solved inst.	480	480	480
Avg. relocations	18.9	15.0	14.0
Avg. CPU time [s]	<0.1	<0.1	<0.1

Strategy	S3-3	S4-3	S5-3	S6-3	S7-3	S8-3
Solved inst.	480	480	480	480	480	460
Avg. relocations	14.9	16.1	13.2	13.8	16.8	13.6
Avg. CPU time [s]	24.8	11.4	11.4	12.1	12.7	66.4
Avg. time per iter.	1.2	1.4	1.4	1.5	1.6	8.0

Strategy	S3-5	S4-5	S5-5	S6-5	S7-5
Solved inst.	480	480	480	480	480
Avg. relocations	13.1	14.6	12.5	13.2	15.0
Avg. CPU time [s]	28.1	10.6	10.8	12.9	11.4
Avg. time per iter.	1.5	2.2	2.2	2.6	2.3

Strategy	S3-7	S4-7	S5-7	S6-7	S7-7
Solved inst.	480	480	480	479	480
Avg. relocations	12.4	13.6	12.2	12.7	13.7
Avg. CPU time [s]	38.3	15.9	22.6	35.5	18.4
Avg. time per iter.	2.2	4.4	6.3	9.7	5.0

SX-Y represents service strategy X with look-ahead horizon Y

S1 to S8 for different look-ahead horizons D. Strategy SX-Y refers to strategy X with look-ahead horizon Y. With strategies S1, S2, S3, S4, S5 and S7 all instances can be solved. One instance cannot be solved sith S6-7 and 20 instances cannot be solved with S8-3.

Run times for strategies S1, S2-1 and S2-2 are fast enough to be applied in real-time at the terminal. For strategies S3 to S7, run times per iteration increase for bigger look-ahead horizons since the underlying models get bigger. But, for bigger look-ahead horizons fewer iterations are necessary and the total run time may decrease. If the truck travel time between the gate and the (un)loading area may be used to determine relocation moves, run times per iteration for strategies S3 to S7 are also sufficient.

The numbers of relocations for all service strategies for different look-ahead horizons are presented in more detail in Figure 10.3. It also compares the dynamic results to the offline solution (Off) where the entire retrieval sequence is known in advance. The x-axis states the service strategy with the associated look-ahead horizon. The y-axis indicates the number of relocations. The boxplots represent the number of relocations obtained for 459 instances (those solved by all strategies). Every boxplot indicates the median, the upper and lower quartiles and outliers for one relocation strategy.

Relocation strategies S2 to S7 with limited knowledge on future retrievals perform well. They outperform the random relocation strategy S1, but cannot reach the solution quality of the offline solution with complete knowledge. For each strategy, the number of relocations and their variances decrease when the look-ahead horizon increases.

Comparing strategies S3 and S4 suggests that it is beneficial to update relocation decisions every time new information becomes available. However, results of S1, S2, S3 and S6 show that for little information ($D \leq 3$) seeking a leveled bay layout may be more beneficial than updating relocation moves; for more information ($D = 5$, $D = 7$) results are similar.

Results for S1, S2 and S6, also show that the benefit of knowing more than the next 3 retrieval containers is limited. Comparing results S5-3 ($D + D' = 6$) with S3-5 and S3-7 ($D = 5$ and $D = 7$) shows that knowing the exact retrieval order of far-away containers is of little benefit. Results of S7 show that keeping one stack empty decreases the solution quality since containers have to be stacked higher in the remaining stacks. Results of S4-3 and S8-3 show that serving trucks in any order rather than in FIFO order reduces the number of relocations.

10.5 Conclusion

This chapter introduced the dynamic container relocation problem that has not been addressed in literature yet. We introduced the expected value of relocations EVR as an indicator to determine the quality of a bay layout with no information on future retrievals. We proved that EVR is minimal for balanced stack heights.

We presented different relocation strategies for partial knowledge of the retrieval sequence. We compared their solution qualities - indicated via the number of relocations - for different look-ahead horizons. Results were also compared to a random relocation strategy and to the optimal offline solution obtained if the entire retrieval sequence is known in advance. It appeared that relocation strategies perform well and outperform the random

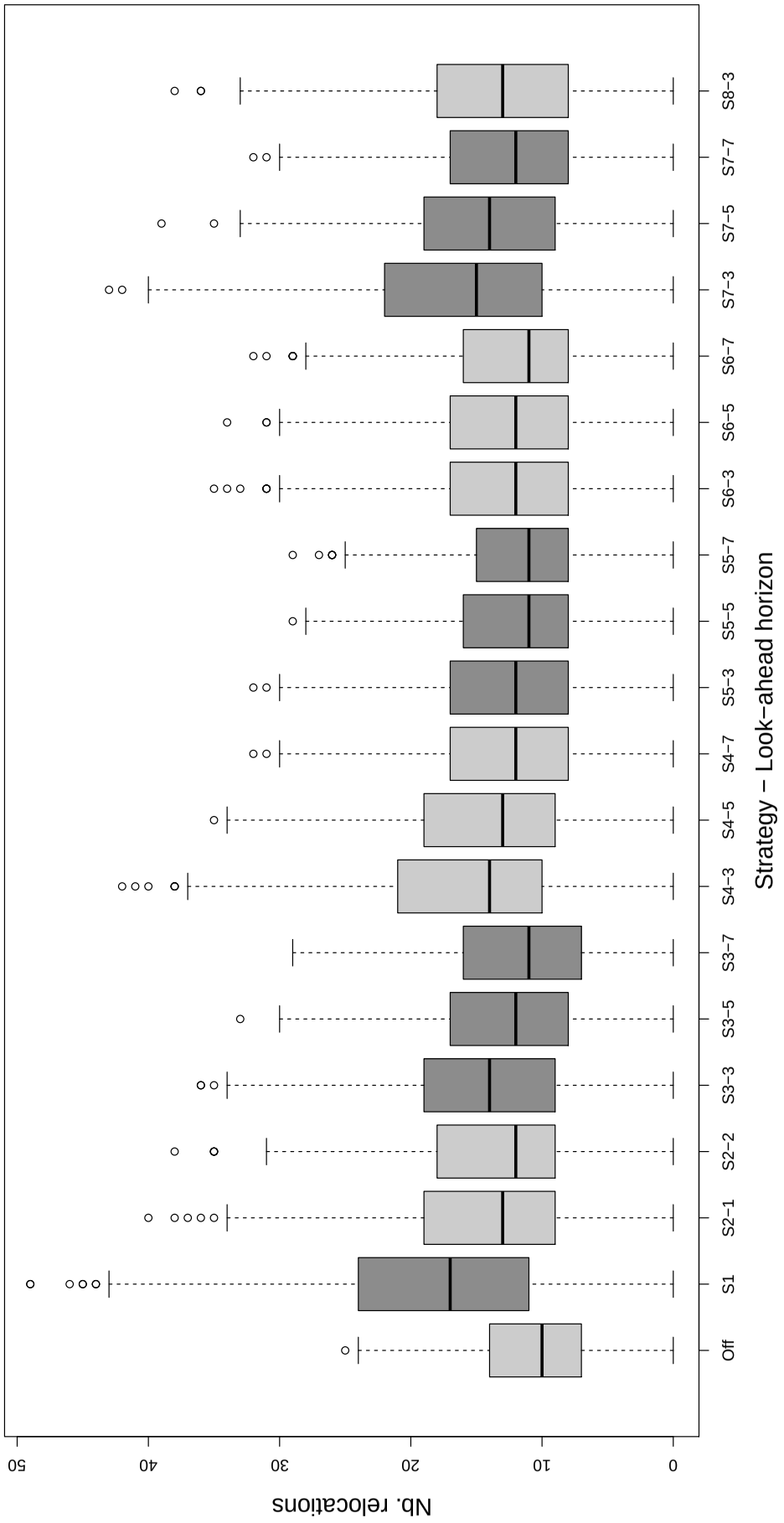


Figure 10.3: Comparison of different relocation strategies for different look-ahead horizons

strategy, but cannot reach the solution quality of the offline solution. Especially, strategies trying to balance stack heights perform well. Run times seem to be short enough to be applied at a terminal in real time.

To continue this work a more generic heuristic for $D > 2$ could be designed. This heuristic could try to balance stack heights and to relocate known retrieval containers using relocation rules from heuristic HC. It would also be interesting to evaluate the competitiveness ratio of the leveling heuristic to obtain more information on the worst case performance of heuristic S2.

For strategies S5 and S6, the layout obtained among those with the same number of relocations depends on cost parameters w_1 and w_2. The obtained layout strongly influences the number of relocations in the subsequent periods since the solution obtained at one iteration fixes the starting layout for the next iteration. It would be interesting to test how the cost parameters influence the solution quality.

It would also be interesting to evaluate the impacts of information on future retrievals and of the point in time when information becomes available (e.g., Wasesa et al.; 2011). This would make it possible to evaluate the potential benefit of new technologies providing the terminal with more details on truck arrivals.

Another approach to tackle the dynamic version would be stochastic programming to include uncertainty directly into the model. The problem can also be extended to deal with dynamic storage and retrieval requests simultaneously. In this case, the problem is to decide in which order to serve trucks, where to locate incoming containers and where to relocate blocking containers in order to minimize truck service times. Little literature exists on this problem (Kim et al.; 2003; Casey and Kozan; 2012).

Conclusion and outlook

Intelligent freight technologies (e.g., EDI, RFID and GPS) monitor and manage physical assets and information flows. Container terminals use these technologies to exchange data with their partners, to locate containers and equipment within the terminal, and to automate tasks. This thesis illustrated, via two examples, how this information may be used to optimize operations at the terminal. In the first part, announced arrival and departure dates and announced volumes are used to optimize the allocation of straddle carriers. In the second part, announced container retrievals and detailed knowledge of container positions are used to optimize retrieval operations. This chapter summarizes the executed work and outlines possibilities to continue this work.

The first part of this thesis addressed the straddle carrier allocation problem (SCAP). The objective is to efficiently allocate straddle carriers to different transport modes (e.g., trucks, trains, barges and vessels). The objective is to minimize overall delays at the terminal. This problem has not been addressed in literature before. The four main contributions of this thesis for the straddle carrier allocation problem are the following. First, we introduced a notation to describe different service strategies applied for different transport modes at different container terminals. We also determined the complexity for some of these service strategies. Second, we represented the straddle carrier allocation problem as a network flow problem: containers to be moved are modeled as flows and available straddle carriers as arc capacities. We modeled the network flow as a generic mixed integer linear program and showed that the generic model can easily be adapted to different service strategies. Third, we carried out a case study for a terminal at the Grand Port Maritime de Marseille. We showed via simulation that results obtained by the deterministic optimization problem remain valid in an uncertain environment. Fourth, we combined the straddle carrier allocation problem with the dimensioning of a truck appointment system. The idea is to use the truck appointment system to deviate truck arrivals to less busy periods. Experiments carried out with the adapted deterministic optimization model and the simulation model showed that this combined approach reduces overall delays at the terminal.

To continue this work, the model may be extended to represent the situation within the terminal in more detail. Container types with their specific requirements could be differentiated. The handling capacity could also be represented in a more detailed way, e.g. by including congestion and container positions within the yard. This would also provide insights on the interconnection between straddle carrier allocation and storage allocation. For this purpose, the optimization model could be combined with simulation or queuing models. It would also be interesting to include stochastic aspects in the optimization model or to combine the straddle carrier allocation problem with human resource management.

The second part of this thesis addressed the container relocation problem (CRP). The objective is to retrieve containers from a bay in a given sequence with a minimum number of parasite relocations. Like most other studies, we considered the case where only containers above the target container may be relocated. The four main contributions of this thesis for the container relocation problem are the following. First, we improved an existing binary program for the container relocation problem. We introduced a preprocessing step - that improves the performance of the binary program considerably - and two new upper bounds on the number of relocations. Second, we presented the first column generation approach for the container relocation problem. We implemented a binary programming subproblem and two variants of an enumerative subproblem. We embedded column generation in a branch and price approach. This approach solves small instances, but does not perform well for bigger instances. Third, we introduced a heuristic branch and price approach. Its objective is to obtain good integer solutions rather than the optimal fractional solution. It uses a heuristic subproblem to generate new columns - based on dual variables and heuristic relocation rules - and runs a heuristic repeatedly to obtain new integer solutions. It obtained good results for some bigger instances, but can probably be improved further. Finally, we addressed the dynamic container relocation problem where the retrieval sequence becomes revealed over time. We introduced a criterion to evaluate the quality of a bay and evaluated different relocation strategies.

This work may be continued in several directions. For a deeper understanding of the problem, domination rules on relocations could be determined. These rules could then be used to obtain more performing heuristics. It would also be interesting to compare the performance of the binary model used here to other existing mixed integer models. It should be worthwhile to search for cuts to add to the binary model and to the master problem. For the exact branch and price approach, a more performing subproblem has to be found to speed up the solution procedure. Alternative branching strategies could also be tested. For the heuristic branch and price approach, the best setting for its parameters has to be determined. In addition, the time spent solving the master problem has to be reduced. The solution quality may also be improved if the solution of the repeatedly run heuristic is improved. For the dynamic container relocation problem, a more theoretic analysis of the problem should be carried out to determine worst case performances. It is also possible to address different variants of the container relocation problem. It would be possible to relax assumption A8 limiting relocations on containers above the target container or to include storage operations. For the dynamic version, different scenarios with regard to the availability and reliability of information could be analyzed.

Appendix A

Extended summary in French

Dans cette thèse, nous utilisons les informations obtenues par les nouvelles technologies (telles que l'échange de données informatisées (EDI), la géolocalisation (GPS) ou la radio-identification (RFID)) pour optimiser les opérations d'un terminal à conteneurs. Nous étudions deux cas : l'affectation de ressources internes dans un terminal et l'enlèvement de conteneurs de la zone de stockage. Ce résumé fait la synthèse du travail présenté dans ce mémoire et met en avant nos différentes contributions.

Chapitre 1 : Introduction générale

Ce chapitre familiarise le lecteur avec le transport conteneurisé. Il décrit le fonctionnement des terminaux à conteneurs et les problèmes d'optimisation existants. De plus, il détaille le contenu du mémoire.

1.1 Transport conteneurisé

Le transport conteneurisé a connu une forte croissance dans les 30 dernières années. Les volumes transportés (mesurés en TEU (équivalent vingt pieds)) ont été multipliés entre 1980 et 2012 pour atteindre 160 millions TEU en 2012. Pour faire face à cette croissance, le nombre et la taille des navires n'ont pas cessé d'augmenter. Le marché du transport conteneurisé maritime est dominé par quelques grands armateurs qui effectuent environ 50% du transport conteneurisé. Les terminaux à conteneurs sont en concurrence pour attirer les armateurs. Etant donné que les coûts d'exploitation des navires sont très importants, la compétitivité d'un terminal à conteneurs dépend surtout des délais d'exécution et des charges pour le (dé)chargement. Mais depuis plusieurs années, améliorer le raccordement d'un terminal à son arrière-pays devient de plus en plus important pour optimiser la chaine de transport globale.

1.2 Terminaux à conteneurs

Les terminaux à conteneurs sont composés de trois zones : le côté maritime où les navires sont (dé)chargés, le yard où les conteneurs sont stockés et le côté terrestre où les camions, les trains et les péniches sont (dé)chargés. Les terminaux utilisent différents types d'équipements. Les grues de quai (dé)chargent les navires. Des cavaliers, des camions ou des véhicules à guidage automatique (AGVs) transportent les conteneurs entre les différentes zones du terminal. Les cavaliers peuvent soulever et poser les conteneurs eux-mêmes, alors que les camions et les AGVs doivent être chargés et déchargés par des grues. Les cavaliers ou des portiques gèrent

la zone de stockage. Dans le premier cas, les cavaliers opèrent le yard et transportent les conteneurs entre les différentes zones. Dans le deuxième cas, les portiques gèrent le yard et un autre type de véhicule transporte les conteneurs.

1.3 Problèmes d'optimisations présents dans les terminaux à conteneurs

La communauté de la recherche opérationnelle trouve une multitude de problèmes d'optimisation dans les terminaux à conteneurs, p.ex. établir le plan de chargement, affecter les postes d'amarrage et les grues de quai aux navires, affecter et ordonnancer le déplacement des conteneurs par les véhicules de transport internes, affecter les conteneurs aux emplacements de stockage. Les approches de résolution exacte ne considèrent souvent qu'un problème spécifique du terminal, alors que les différents problèmes sont fortement reliés. La simulation est souvent utilisée pour représenter le terminal entier et les aspects stochastiques.

1.4 Nouvelles technologies

Les nouvelles technologies permettent de contrôler et de gérer les biens physiques et le flux d'informations. Les terminaux à conteneurs les utilisent pour échanger des données avec les armateurs, les transporteurs et la douane, mais aussi pour localiser les conteneurs et leurs véhicules dans le terminal. Les nouvelles technologies sont aussi indispensables pour l'automatisation. Les nouvelles technologies permettent au terminal d'améliorer la prise de décisions et de renforcer la sécurité.

En communiquant avec les armateurs et les transporteurs, les terminaux à conteneurs obtiennent des informations sur les volumes et les arrivées prévues et peuvent mieux gérer leurs ressources internes. En ayant des informations exactes sur l'emplacement de chaque conteneur, les terminaux peuvent gère la zone de stockage plus efficacement. Le GPS et / ou la RFID sont utilisés pour localiser les conteneurs.

Part I : Problème d'affectation de cavaliers

La première partie de cette thèse traite le problème d'affectation de cavaliers aux véhicules externes. Nous utilisons les informations sur les volumes et les dates d'arrivées prévues pour optimiser l'affectation de cavaliers dans le but de minimiser la globalité des retards au terminal. Cette étude se limite aux terminaux à conteneurs utilisant des cavaliers pour transporter et (dé)stocker des conteneurs. Les cavaliers sont partagés entre différents mode de transport (navire, camion, train, péniche). Le temps de service d'un véhicule externe dépend fortement du nombre de cavaliers qui lui sont affectés. Actuellement, quelques terminaux utilisent des systèmes de rendez-vous pour camions pour réduire les temps de service des camions. Nous analysons les impacts d'un tel système sur la globalité des retards dans le terminal. Nous utilisons la programmation en nombres entiers pour résoudre ce problème.

Chapitre 2 : Problème d'affectation de cavaliers

Le problème d'affectation de cavaliers n'a pas encore été étudié dans la littérature scientifique. Ce chapitre introduit la problématique et résume la littérature reliée.

2.1 Description du problème

Nous considérons un terminal qui n'utilise que des cavaliers pour les tâches de transport et de stockage et qui sert différents modes de transport comme les navires, les camions, les trains et les péniches. Nous supposons que le terminal affecte un cavalier à un type de tâche pour une certaine durée pour simplifier la tâche des opérateurs. Cela a pour conséquence que l'affectation des cavaliers ne peut être modifiée qu'à intervalles discrets. Les cavaliers sont conduits par des opérateurs. Le terminal peut ainsi adapter sa capacité d'un jour à l'autre par le nombre d'opérateurs embauchés. L'horizon du temps du problème d'affectation de cavaliers est d'une journée de travail. Nous nous intéressons à deux questions : 1) Combien de cavaliers faut-il pour le lendemain ? 2) Comment affecter ces cavaliers aux différents modes de transport ? L'objectif est de réduire le retard de tous les modes de transport pour augmenter la compétitivité du terminal.

L'affectation optimale dépend des conteneurs à (dé)charger pour chaque véhicule externe. Les informations obtenues sur les arrivées et les volumes des navires, des trains et des péniches sont assez fiables. Les arrivées et les volumes des camions peuvent être estimés, mais ne sont pas connus avec certitude. Nous traitons ce problème à un niveau tactique et supposons que l'ordonnancement de conteneurs sur les cavaliers est fait en temps réel par le système opérationnel du terminal.

Nous divisons la journée en T périodes car les cavaliers ne peuvent être réaffectés qu'à intervalles discrets. Nous utilisons le terme 'tâche' pour désigner toutes les opérations qui doivent être exécutées pour chaque conteneur traité dans le terminal. Le terminal possède des informations sur l'ensemble des véhicules arrivant I, la date d'arrivée r_i et le nombre de tâches par véhicule p_i. Toutes les tâches doivent être exécutées avant la fin de la journée. La capacité du terminal dépend du nombre de cavalier s_t par période t et de leur débit. Nous définissons ce débit par un nombre moyen de tâches h qu'un cavalier peut exécuter dans une période.

Les différents terminaux à conteneurs servent les différents modes de transport avec des stratégies différentes. La stratégie choisie dépend du volume, du coût d'exploitation, du niveau de connaissance et de la fiabilité des informations et de l'équipement du terminal. Nous introduisons une notation $\alpha|\beta|\gamma$ pour décrire la stratégie utilisée pour un mode de transport. α indique si un cavalier est affecté à exactement un véhicule ou s'il peut être partagé entre tous les véhicules du même mode de transport. β décrit des spécifications additionnelles pour un mode de transport comme la date d'arrivée, la date de départ, le débit maximal. γ décrit les différentes manières de mesurer le retard, comme le temps passé au terminal ou le nombre de tâches qui ne sont pas exécutées au départ du véhicule. La figure A.1 expose cette problématique.

2.2 Littérature

Ce chapitre résume la littérature reliée au problème d'affectation de ressources. Peu d'articles optimisent l'affectation de ressources dans les terminaux à conteneurs (Gambardella et al.; 1998, 2001; Vis et al.; 2005; Alessandri et al.; 2008; Kang et al.; 2008). Ces articles ont pour but de minimiser les temps de service des navires, mais négligent le côté terrestre du terminal. Notre approche est plus vaste car on considère aussi bien le côté maritime que le côté terrestre du terminal.

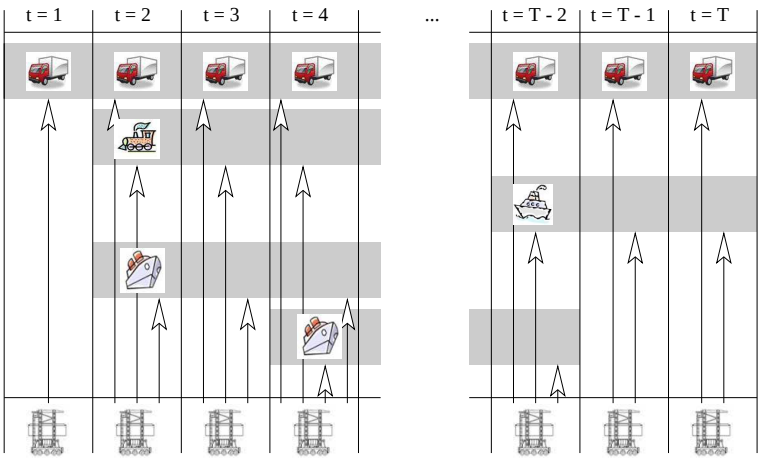


Figure A.1: Problème d'affectation de ressources

Plusieurs articles s'adressent au problème opérationnel d'ordonnancement des conteneurs sur les ressources internes disponibles (Kim and Kim; 1999; Böse et al.; 2000; Das and Spasovic; 2003; Hartmann; 2004; Kim and Bae; 2004; Bish et al.; 2005; Briskorn et al.; 2006; Froyland et al.; 2008; Balev et al.; 2009; Nguyen and Kim; 2009, 2010; Lee et al.; 2010; Skinner et al.; 2013). Cette problématique est très différente de la nôtre. Premièrement, ils se basent sur un nombre de ressources disponibles donné, tandis que nous essayons de déterminer le nombre de ressources à affecter à chaque mode de transport. Deuxièmement, ils se placent à un niveau opérationnel et supposent que l'on connait l'heure exacte à laquelle chaque conteneur doit être chargé, alors que nous nous plaçons au niveau tactique où nous avons pas ces informations détaillées. Souvent ces articles se focalisent sur le côté maritime du terminal.

D'autres études combinent l'ordonnancement des tâches avec la planification de la main d'œuvre (Legato and Monaco; 2003; Fancello et al.; 2011; Kim et al.; 2004) ou l'affectation des emplacements de stockage (Kozan and Preston; 2006; Hadjiconstantinou and Ma; 2009; Lee et al.; 2009; Wu et al.; 2013). D'autres articles, enfin, ordonnancent différents types de véhicules internes simultanément (Meersmans and Wagelmans; 2001; Chen et al.; 2007; Lau and Zhao; 2008; Zeng and Yang; 2009; Cao et al.; 2010; Chen, Langevin and Lu; 2013).

Chapitre 3 : Problème d'affectation de cavaliers

Ce chapitre représente le problème d'affectation de cavaliers comme un problème de flot et le modèlise sous la forme d'un programme linéaire mixte. Nous présentons un modèle général et des extensions pour adapter ce modèle à différentes stratégies de service. Nous discutons aussi de la complexité des différentes stratégies.

3.1 Modèle de flot général

Nous représentons la problématique d'affectation de cavaliers comme un problème de flot. Le flot représente les conteneurs à transporter et les capacités des arcs la capacité des cavaliers affectés. Cette modélisation est inspirée par Gambardella et al. (2001). La figure A.2 illustre cette formulation. Elle représente un terminal qui sert deux véhicules pendant une journée de travail. Les nœuds ronds représentent les périodes de la journée. Les nœuds rectangulaires sont des sources de flot et représentent l'arrivée des véhicules avec leurs tâches p_i^m.

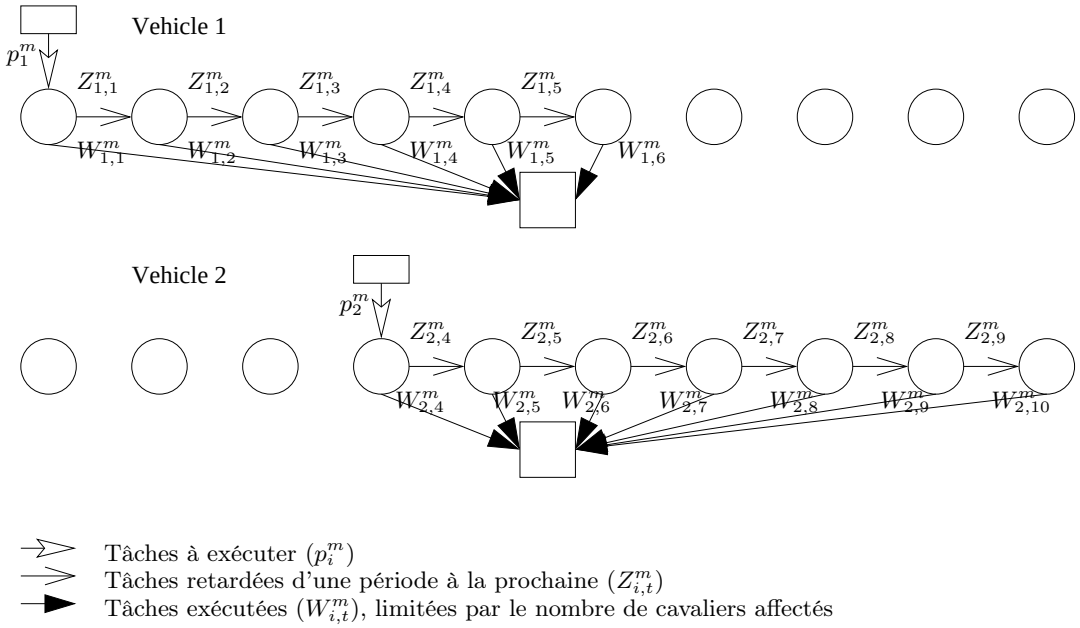


Figure A.2: Modèle de flot général

Chaque source est reliée à la période qui correspond à l'arrivé du véhicule. Les nœuds carrés représentent les puits. Les flots $W_{i,t}^m$ sur les arcs reliant les périodes avec les puits représentent les tâches exécutées par véhicule et par période. Les capacités sur les arcs limitent le nombre de tâches qui peuvent être exécutées en fonction du nombre de cavaliers affectés. Les flots $Z_{i,t}^m$ sur les arcs reliant deux périodes représentent les tâches qui ne sont pas exécutées et retardées d'une période. Il n'est pas possible de retarder des tâches au-delà du départ du véhicule.

Nous modélisons le problème de flot comme un programme linéaire mixte. Nous utilisons les paramètres et les variables suivants :

T	Nombre de périodes de temps de la journée
M	Nombre de mode de transports connectés au terminal
I^m	Nombre de véhicules du mode de transport m qui arrivent au terminal
t	Index d'une période de temps, $t = 1, \ldots, T$
m	Index d'un mode de transport, $m = 1, \ldots, M$
i	Index d'un véhicule du mode de transport m, $i = 1, \ldots, I^m$
r_i^m	Période dans laquelle le véhicule i du mode de transport m arrive au terminal
d_i^m	Période dans laquelle le véhicule i du mode de transport m part du terminal
s_t	Nombre de cavaliers disponibles à la période t
h^m	Nombre moyen de tâches qu'un cavalier peut effectuer par période pour le mode de transport m
$X_{i,t}^m$	Nombre de cavaliers affectés au véhicule i du mode de transport m à la période t
$W_{i,t}^m$	Nombre de tâches exécutées par le véhicule i du mode de transport m à la période t
$Z_{i,t}^m$	Nombre de tâches du véhicule i du mode de transport m qui sont transférées à la période $t+1$

Nous modélisons le modèle générique où chaque véhicule doit être servi dans sa fenêtre

de temps. Ici, nous ne représentons que les contraintes les plus importantes.

$$W_{i,t}^m \leq h^m \cdot X_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = 1,\ldots,T \tag{3.1}$$

$$Z_{i,t}^m = p_i^m - W_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = r_i^m \tag{3.2}$$

$$Z_{i,t}^m = Z_{i,t-1}^m - W_{i,t}^m \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m, t = r_i^m + 1,\ldots,T \tag{3.3}$$

$$Z_{i,d_i^m}^m = 0, \qquad \forall m = 1,\ldots,M, i = 1,\ldots,I^m \tag{3.4}$$

$$\sum_{m=1}^{M} \sum_{i=1}^{I^m} X_{i,t}^m \leq s_t \qquad \forall t = 1,\ldots,T \tag{3.5}$$

La contrainte (3.1) impose que le nombre de tâches exécutées par véhicule et par période ne dépasse pas la capacité des cavaliers affectés. Les contraintes (3.2) et (3.3) sont les contraintes de flot pour les tâches arrivées, exécutées et retardées. La contrainte (3.4) impose que chaque véhicule est entièrement servi avant sa date de départ. La contrainte (3.5) limite le nombre de cavaliers affectés par le nombre de cavaliers disponibles.

Grâce à la modularité du modèle nous pouvons simplement représenter différents terminaux avec différentes stratégies de service. Nous créons un sous-modèle pour chaque mode de transport qui représente ses caractéristiques. Pour ceci nous étendons le modèle générique avec les éléments présentés dans la section 3.2. Les différents sous-modèles sont ensuite reliés par une contrainte qui limite le nombre total de cavaliers affectés.

3.2 Extensions du modèle pour représenter différentes stratégies

Nous ajoutons des paramètres, variables et contraintes pour adapter le modèle générique à différentes stratégies de service. Nous implémentons les cas ci-dessous. Nous indiquons aussi quelle notation nous utilisons pour décrire chaque élément.

- Affectation de cavaliers (α)
  - Les cavaliers sont affectés à un véhicule ($\alpha = ded$)
  - Les cavaliers sont affectés à un mode de transport et partagés parmi tous les véhicules du même mode de transport ($\alpha = shar$)
- Contraintes additionnelles (β)
  - Les tâches ne peuvent pas être exécutées avant l'arrivée du véhicule ($\beta = r_v$)
  - Les tâches ne peuvent pas être exécutées après le départ du véhicule ($\beta = d_v$)
  - Le nombre maximal de tâches qui peuvent être exécutées par véhicule et par période est limité ($\beta = \max_v$)
  - Le nombre maximal de tâches qui peuvent être exécutées par mode de transport et par période est limité ($\beta = \max_m$)
  - Le nombre de cavaliers affectés ne peut pas être augmenté ($\beta = non - incr$)
- Mesures de retard (γ)

- Minimiser le temps de service ($\gamma = \sum C_v$)
- Minimiser le temps de service après la date de fin souhaitée ($\gamma = \sum T_v$)
- Minimiser le nombre de tâches non-exécutées ($\gamma = \sum U_c$)
- Servir le véhicule dans sa fenêtre de temps ($\gamma = -$)
- Minimiser le nombre d'équipes qui servent un véhicule ($\gamma = S_v$)

3.3 Analyse de sensibilité

Cette section présente une analyse de sensibilité. Dans un premier temps, nous déterminons la meilleure implémentation du problème de flot comme programme linéaire mixte. Puis, nous évaluons l'impact de différents paramètres. Nous nous apercevons que le nombre de véhicules, le nombre de périodes, la largeur des fenêtres de temps et la stratégie appliquées ont le plus d'impact sur le temps de calcul.

3.4 Analyse de complexité

Nous démontrons que notre problème d'affectation généralise des problèmes d'ordonnancement. Ceci nous permet d'obtenir la complexité de différentes stratégies, qui peuvent être réduites, pour la plupart, à un problème d'ordonnancement avec une complexité connu.

3.5 Formulation alternative

Nous présentons une formulation qui représente tous les véhicules du même mode de transport de manière agrégée. Cette modélisation peut être utilisée seulement si les cavaliers sont partagés parmi tous les véhicules du même mode de transport et ne peut donc être utilisée que pour une partie des stratégies. Nous remplaçons les variables représentant un véhicule (p_i^m, $X_{i,t}^m$, $W_{i,t}^m$ et $Z_{i,t}^m$) par des variables représentant un mode de transport avec tous ses véhicules (p_t^m, X_t^m, W_t^m et Z_t^m). La figure A.3 représente le modèle agrégé pour un mode de transport. Pour le modèle agrégé la taille du problème dépend seulement du nombre de périodes et ne dépend plus du nombre de véhicules. Des expérimentations démontrent que le modèle agrégé est plus performant que le modèle non-agrégé.

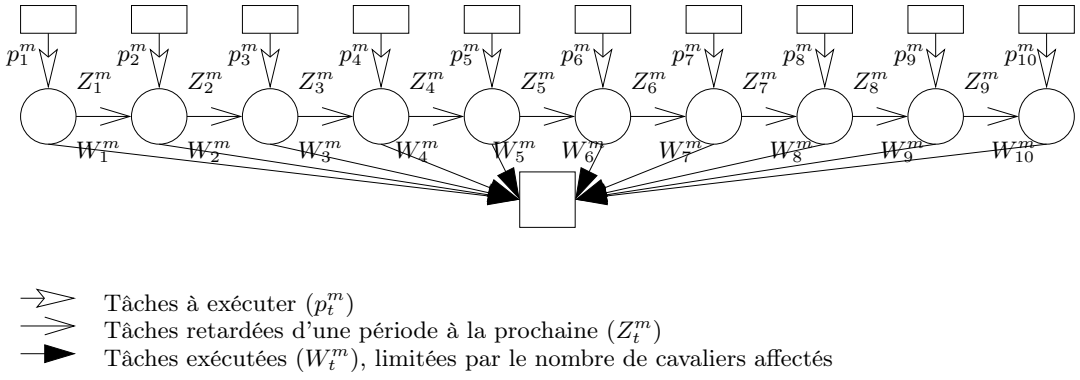


Figure A.3: Modèle de flot agrégé

3.6 Perspectives

Cette section présente des perspectives pour continuer ce travail. Notre modèle pour l'affectation des cavaliers inclut les caractéristiques les plus importantes. Il serait possible de le rendre encore plus réaliste en rajoutant des détails comme les différents types de conteneurs ou les accords syndicaux. Une autre possibilité serait de combiner notre modèle avec de la simulation ou des systèmes de files d'attente pour modéliser la capacité des cavaliers en fonction du nombre de cavalier affectés et de l'emplacement des véhicules externes et des conteneurs dans le yard. Le modèle déterministe ne prend pas en compte tous les aléas qui apparaissent dans des terminaux à conteneurs.

4. Etude de cas pour le Grand Port Maritime de Marseille

Nous effectuons une étude de cas pour un terminal du Grand Port Maritime de Marseille (GPMM). Nous appliquons notre modèle d'optimisation sur des données réelles. Nous démontrons via la simulation que l'affectation obtenue par le modèle d'optimisation se comporte bien dans un contexte stochastique et que les retards sont estimés correctement.

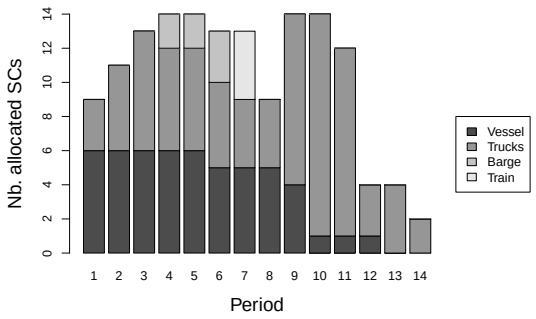
4.1 La situation à Marseille

GPMM est un port généraliste qui est spécialement actif dans le domaine pétrolier. Il est l'un des plus grands ports en Europe et le plus grand port en France. En 2012, il a manipulé 1 060 00 TEU. Le terminal étudié est un terminal multimodal qui sert des navires, des camions, des trains et des péniches. Les navires doivent être servis dans leur fenêtre de temps, le débit est limité par le débit des grues de quai. Les cavaliers sont affectés à un véhicule et le nombre de cavaliers affectés ne peut pas être augmenté. $(ded|r_v, d_v, non-incr, \max_v|\sum C_v)$. Les barges doivent être servies le plus vite possible, le débit est limité par le débit des grues de quai et les cavaliers sont affectés à un véhicule $(ded|r_v, d_v, \max_v|\sum C_v)$. Les trains quittent le terminal à une heure fixée même si toutes les tâches ne sont pas exécutées et les cavaliers sont partagés parmi tous les trains $(shar|r_v, d_v, |\sum U_c)$. Les camions doivent être servis le plus tôt possible mais au plus tard avant la fin de la journée. Les cavaliers sont partagés entre tous les camions. $(shar|r_v, d_v = T, |\sum C_v)$. Nous présentons aussi les instances utilisées dans la suite extraites de données obtenues du terminal.

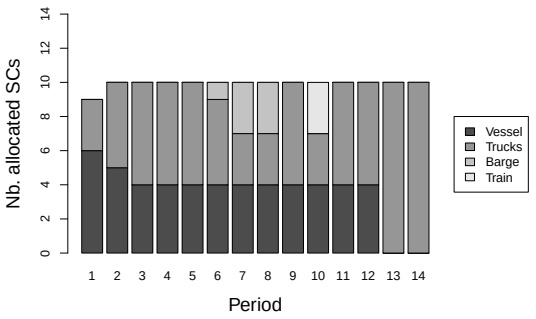
4.2 Modèle d'optimisation

Nous combinons les éléments présentés dans le Chapitre 3 pour modéliser le terminal avec les stratégies appliquées pour les navires, les camions, les trains et les péniches. Pour les navires, les trains et les péniches chaque véhicule est représenté indépendamment. Les camions sont représentés de manière agrégée.

Nous résolvons chaque instance avec différents nombres de cavaliers disponibles. Tous les scénarios sont résolus très rapidement (en moins d'une seconde). La solution obtenue nous indique comment affecter les cavaliers aux différents véhicules et les retards qui résultent de cette affectation. La figure A.4 montre l'affectation et les retards associés pour une instance avec 14 et 10 cavaliers disponibles. Pour déterminer le nombre de cavaliers nécessaires pour le lendemain l'opérateur du terminal peut choisir le nombre minimum de cavaliers pour lequel il obtient la qualité de service souhaitée.



(a) 14 cavaliers : 1 tâche camion retardée, barge servies dans 3 périodes, 0 tâches de train non-exécutées



(b) 10 cavaliers : 607 tâches camion retardées, barge servies dans 5 périodes, 2 tâches de train non-exécutées

Figure A.4: Allocation optimale et retards associés

4.3 Modèle de simulation

Nous décrivons le modèle de simulation utilisé pour représenter les aspects stochastiques du terminal. Le modèle reproduit les stratégies appliquées pour les différents modes de transport. De plus, il tient compte des interactions entre les différents véhicules, les cavaliers et les grues de quai et rajoute de l'aléa sur l'arrivée et le volume des véhicules et sur le temps de transport des conteneurs.

Pour les expérimentations nous adaptons le modèle de simulation sous Arena proposé par Rodriguez Verjan and Dauzère-Pérès (2010). Le modèle de simulation prend l'affectation des cavaliers déterminée par le modèle d'optimisation comme entrée. Nous évaluons cette affectation pour différents niveaux de variabilité. Les résultats montrent que les retards estimés par le modèle d'optimisation sont corrélés avec les retards obtenus par le modèle de simulation.

Chapitre 5 : Combinaison avec le dimensionnement d'un système de rendez-vous pour camions

Les terminaux à conteneurs utilisent parfois un système de rendez-vous pour réduire la congestion au terminal. L'idée est d'étaler l'arrivée des camions sur la journée pour réduire la congestion aux heures de pointe. Les systèmes de rendez-vous permettent aux opérateurs du terminal de mieux prévoir la charge journalière et d'adapter les opérations. Les entreprises de transport routier profitent de temps de service réduits (Sgouridis and Angelides; 2002; Srour et al.; 2003; Morais and Lord; 2006; Giuliano and O'Brien; 2007; Maguire et al.; 2010). Dans ce chapitre nous combinons l'affectation des cavaliers avec le dimensionnement d'un système de rendez-vous pour camion. Nous étudions les impacts d'un système de rendez-vous pour camions sur le retard global du terminal.

5.1 Introduction

Les systèmes de rendez-vous limitent le nombre de camions qui peuvent rentrer dans le terminal par créneau horaire. Différentes manière d'implémenter ces systèmes existent (Morais and Lord; 2006; Giuliano and O'Brien; 2007)). Nous nous concentrons sur le cas où la prise

de rendez-vous est obligatoire pour entrer dans le terminal. Le rendez-vous doit être pris pour un conteneur et pour un créneau. En retour, le terminal s'engage à servir chaque camion dans le créneau réservé. Pour limiter l'impact sur les camions, les créneaux proposés doivent être le plus proche possible des créneaux souhaités. Notre but est d'utiliser le système de rendez-vous pour ainsi déplacer l'arrivée des camions dans des périodes moins chargées et réduire le retard des camions, des trains, des péniches et des navires.

Peu de littérature existe sur le dimensionnement des systèmes de rendez-vous (Murty et al.; 2005; Ioannou et al.; 2006; Huynh and Walton; 2008; Guan and Liu; 2009; Chen et al.; 2011; Chen, Govindan and Yang; 2013). Ces études supposent que la capacité affectée aux camions est donnée et déterminent le nombre de camions à accepter pour cette capacité. Ils négligent que les ressources internes sont partagées entre différents modes de transport.

5.2 Modèle linéaire

Nous adaptons le modèle agrégé du Chapitre 3 pour représenter le système de rendez-vous. La figure A.5 représente le nouveau modèle. Les flots p_r^m représentent le nombre de tâches qui souhaitent arriver à la période r. Les flots $Z_{r,t}^m$ représentent le nombre de tâches qui souhaitent arriver à la période r, mais qui sont affectées à la période t. Les flots W_t^m représentent le nombre de tâches à exécuter dans la période t et par conséquent le nombre de rendez-vous à offrir pour la période t. Comme avant, W_t^m est limitée par la capacité des cavaliers affectés.

Pour représenter le terminal complet nous combinons ce sous-problème avec les sous-problèmes des autres modes de transport. Ceci nous permet de minimiser le retard global du terminal.

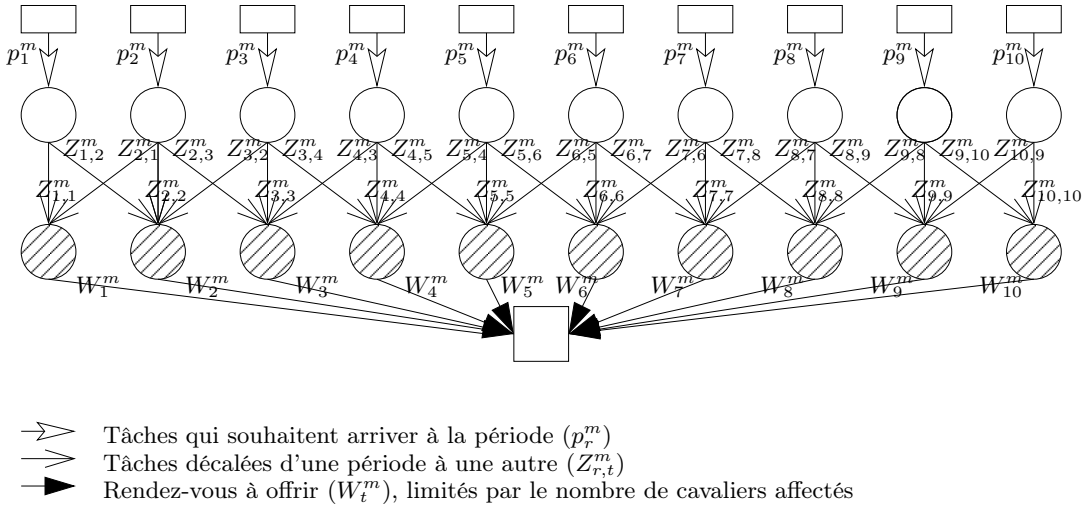


Figure A.5: Modèle de flot pour le dimensionnement d'un système de rendez-vous pour camions

5.3 Expérimentations numériques

Nous exécutons ce modèle avec une déviation maximale de 1 et de 2 périodes. Nous comparons les retards obtenus avec les résultats obtenus sans système de rendez-vous. Les résultats montrent que cette approche intégrée décale l'arrivée des camions vers des créneaux

moins chargés. Ceci permet d'utiliser les cavaliers plus efficacement et de diminuer le retard des camions, des trains et des péniches. Le système de rendez-vous pour camions peut alors réduire le délai global au terminal.

Des expérimentations avec le système de simulation démontrent enfin que ces conclusions restent valables dans un environnement stochastique.

Part II : Problème de repositionnement de conteneurs

La deuxième partie de la thèse traite le problème de repositionnement de conteneurs. Nous utilisions les informations sur les conteneurs à enlever et leurs emplacements pour optimiser le processus d'enlèvement. Les terminaux à conteneurs empilent les conteneurs pour mieux utiliser la surface disponible. L'inconvénient est que seulement le conteneur du haut peut être enlevé directement. Si un autre conteneur doit être enlevé, les conteneurs bloquants doivent être repositionnés. Le but du problème de repositionnement de conteneurs est d'enlever toutes les conteneurs d'une rangée dans l'ordre indiqué en minimisant les repositionnements parasites. Nous nous intéressons à la version statique et à la version dynamique du problème. Nous proposons différentes approches de type branch and price pour ce problème.

Chapitre 6 : Problème de repositionnement de conteneurs

Le problème de repositionnement de conteneurs existe dans la littérature scientifique sous le nom de « container relocation problem (CRP) ». Ce chapitre présente la problématique et résume la littérature associée.

6.1 Description du problème

Pour découpler les opérations maritimes et terrestres, les conteneurs entrants ne sont pas immédiatement chargés sur un véhicule sortant, mais sont stockés dans le yard pendant plusieurs jours. Pour mieux utiliser l'espace disponible, les terminaux empilent les conteneurs. Par conséquent, seul le conteneur en haut d'une pile est directement accessible. Si un autre conteneur doit être enlevé, les conteneurs dessus doivent être repositionnés. Les terminaux essayent d'éviter ces mouvements improductifs car ils nuisent à la performance globale du terminal. Toutefois, les repositionnements ne peuvent pas être évités complètement car peu d'informations sur les enlèvements futurs sont connues quand un conteneur doit être stocké.

Le nombre de repositionnements croît avec la hauteur des piles de conteneurs et est un problème plus important dans les terminaux qui utilisent des cavaliers dans le yard. La figure A.6 représente un tel terminal et montre comment le yard est divisé en plusieurs blocs, chaque bloc en plusieurs rangées, chaque rangée en plusieurs piles et chaque pile en plusieurs niveaux. Grâce aux nouvelles technologies, le terminal sait exactement à quelle position (bloc, rangée, pile, niveau) chaque conteneur est stocké et quelles positions sont vides.

Nous étudions le problème de repositionnement de conteneurs tel qu'il est défini dans la littérature académique. Le plan de chargement des navires et l'ordre de service des camions sont connus et imposent l'ordre d'enlèvement des conteneurs.

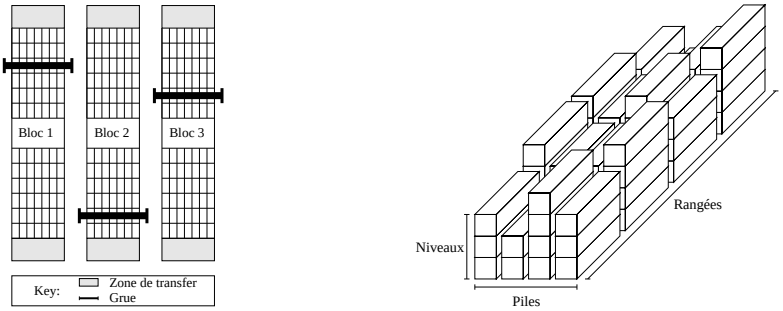


Figure A.6: Blocs, rangées, piles et niveaux

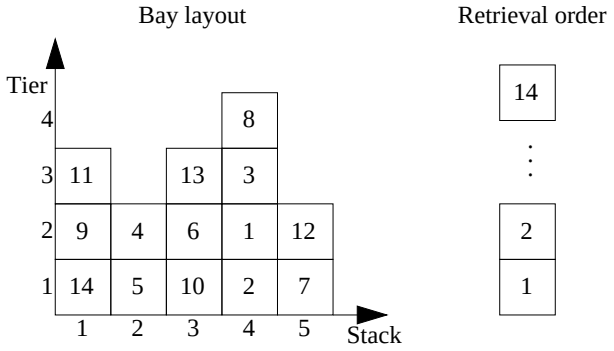


Figure A.7: Problème de repositionnement de conteneurs

En règle générale, la configuration de la rangée ne correspond pas à l'ordre d'enlèvement et des conteneurs doivent être repositionnés. La figure A.7 illustre la problématique. L'objectif est d'enlever tous les conteneurs dans l'ordre donné avec le minimum de repositionnements. Deux variantes du problème existent : tous les conteneurs peuvent être repositionnés ou seulement les conteneurs au-dessus du conteneur cible. La définition du problème repose sur des hypothèses A1 à A7 et inclut ou n'inclut pas l'hypothèse A8.

- A1: La rangée initiale et l'ordre d'enlèvement des conteneurs ou des groupes de conteneurs sont connus à l'avance.
- A2: Aucun conteneur n'arrive pendant le processus d'enlèvement
- A3: Seul le conteneur en haut d'une pile peut être enlèvé. Un conteneur ne peut être repositionné qu'en haut d'une pile ou sur une pile vide.
- A4: Les conteneurs ne sont repositionnés que dans la même rangée car des repositionnements entre rangées sont trop lents.
- A5: La rangée est limitée par le nombre maximal de piles et de niveaux.
- A6: Tous les conteneurs dans la même rangée ont la même taille et peuvent être empilés dans n'importe quel ordre.
- A7: La distance parcourue à l'intérieur d'une rangée (horizontalement et verticalement) a peu d'impact sur le temps d'exécution.
- A8: Seulement les conteneurs dits bloquants, situés au-dessus du conteneur cible, peuvent être repositionnés.

Comme la plupart des études, nous abordons ce problème avec des contraintes de priorité sur des conteneurs individuels et nous incluons l'hypothèse A8.

MIP BB other

6.2 Littérature

Le problème de repositionnement de conteneurs est NP-difficile (Caserta et al.; 2012). Peu d'approches exactes existent, toutes basées sur des programmes linéaires mixtes (Lee and Hsu; 2007; Caserta et al.; 2012; Petering and Hussein; 2013; Tang et al.; 2012). D'autres études utilisent des méthodes heuristiques pour résoudre le problème. Ces approches sont souvent de type branch and bound heuristique (Kim and Hong; 2006; Wu and Ting; 2010; Zhang et al.; 2010; Forster and Bortfeldt; 2012; Rei and Pedroso; 2012; Ünlüyurt and Aydin; 2012; Zhu et al.; 2012). Mais d'autres approches existent : basée sur la recherche locale, comme la recherche tabou, ou à base de programmation dynamique (Caserta et al.; 2009; Caserta and Voß; 2009; Wu et al.; 2009; Caserta, Voß and Sniedovich; 2011).

6.3 Bornes sur le nombre de relocations

Dans la littérature, existent déjà une borne supérieure et une borne inférieure sur le nombre de repositionnement nécessaires pour enlever tous les conteneurs. La borne supérieure est obtenue grâce à une heuristique gloutonne et la borne inférieure à l'aide de l'état initial de la rangée. Nous utilisons ces deux bornes pour introduire une nouvelle borne sur le nombre de relocations maximales par enlèvement et une condition d'optimalité.

6.4 Instances

Cette section présente les instances introduites par Caserta, Voß and Sniedovich (2011) et discute de leurs difficulté. Ces instances sont fréquemment utilisées pour comparer les différentes approches de résolution et nous allons évaluer la qualité de nos approches sur ces instances.

Chapitre 7 : Programmes binaires

Ce chapitre présente un programme binaire existant mais incorrect. Nous allons corriger et améliorer ce programme. En particulier, nous rajoutons une procédure de prétraitement pour fixer plusieurs variables et des coupes. Nous présentons des résultats expérimentaux pour évaluer la qualité du prétraitement et des coupes.

7.1 Modèle de Caserta et al.

Caserta et al. (2012) introduisent des variables de configuration et des variables de mouvements :

$$b_{ijnt} = \begin{cases} 1 & \text{si le conteneur } n \text{ se trouve à la position } (i,j) \text{ (pile } i\text{, niveau } j\text{)} \\ & \text{au début de la période } t, \\ 0 & \text{sinon;} \end{cases}$$

$$x_{ijklnt} = \begin{cases} 1 & \text{si le conteneur } n \text{ est repositionné de la position } (i,j) \text{ à la position } (k,l) \\ & \text{pendant la période } t, \\ 0 & \text{sinon;} \end{cases}$$

$$y_{ijnt} = \begin{cases} 1 & \text{si le conteneur } n \text{ est enlevé de la position } (i,j) \text{ pendant la période } t \\ 0 & \text{sinon.} \end{cases}$$

Ils utilisent ces variables pour formuler le modèle binaire. Ce modèle minimise le nombre de relocations et impose les contraintes physiques et la continuité dans le temps. Mais la contrainte qui impose l'ordre LIFO est trop restrictive et les repositionnements ne sont pas limités aux conteneurs en dessus du conteneur à enlever.

7.2 Améliorations du modèle

Nous formulons des contraintes qui imposent l'ordre LIFO et limitent les conteneurs qui peuvent être repositionnés correctement. De plus, nous enlevons des variables et des contraintes superflues. Nous proposons le modèle présenté ci-dessous.

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{t=1}^{T-1}\sum_{n=t+1}^{N} x_{ijklnt}$$

s.t.

$$\sum_{n=t}^{N} b_{ijnt} \leq 1 \qquad \forall i = 1,\dots,W, j = 1,\dots H, t = 1,\dots,T-1 \tag{7.16}$$

$$\sum_{n=t}^{N} b_{ijnt} \geq \sum_{n=t}^{N} b_{ij+1nt} \qquad \forall i = 1,\dots,W, j = 1,\dots H-1, t = 1,\dots,T-1 \tag{7.17}$$

$$b_{ijnt+1} = b_{ijnt} + \sum_{k=1}^{W}\sum_{l=1}^{H} x_{klijnt} - \sum_{k=1}^{W}\sum_{l=1}^{H} x_{ijklnt} \qquad \forall i = 1,\dots,W, j = 2,\dots H, t = 1,\dots,T-2, n = t+1,\dots,N \tag{7.18}$$

$$b_{ijnt+1} = b_{ijnt} - y_{ijtt} \qquad \forall i = 1,\dots,W, j = 1,\dots H, t = 1,\dots,T-2, n = t \tag{7.19}$$

$$\sum_{i=1}^{W}\sum_{j=1}^{H} y_{ijtt} = 1 \qquad \forall t = 1,\dots,T-1 \tag{7.20}$$

$$M \cdot \left(1 - \sum_{n=t+1}^{N} x_{ijklnt}\right) \geq \sum_{n=t+1}^{N}\sum_{j'=j+1}^{H}\sum_{l'=l+1}^{H} x_{ij'kl'nt} \qquad \forall i = 1,\dots,W, j = 2,\dots H-1, k = 1,\dots,W, l = 1,\dots H-1, t = 1,\dots,T-1 \tag{7.21}$$

$$M \cdot \sum_{j=1}^{H} y_{ijtt} \geq \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{n=t+1}^{N} x_{ijklnt} \quad \forall i = 1, \ldots, W, t = 1, \ldots, T-1 \tag{7.22}$$

$$M \cdot y_{ijtt} + \sum_{j'=2}^{j} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{n=t+1}^{N} x_{ij'klnt} \leq M \quad \forall i = 1, \ldots, W, j = 3, \ldots H, t = 1, \ldots, T-1 \tag{7.23}$$

La fonction objectif minimise le nombre de repositionnements. Les contraintes (7.16) et (7.17) imposent que chaque position est occupée par au plus un conteneur et qu'il n'y a pas conteneur flottant. Les contraintes (7.18) et (7.19) garantissent la continuité de la configuration dans le temps par les repositionnements et les enlèvements exécutés. La contrainte (7.20) impose que le conteneur t est enlevé à la période t. La contrainte (7.21) impose l'ordre LIFO. Les contraintes (7.22) et (7.23) limitent les conteneurs qui peuvent être repositionnés.

Nous proposons un prétraitement basé sur la configuration initiale, sur le fait que seulement les conteneurs bloquants peuvent être repositionnés et sur le nombre de conteneurs dans la rangée à chaque période. Ce prétraitement fixe des variables b_{ijnt} et x_{ijklnt} à 0 ou 1. Nous proposons aussi des coupes basées sur les bornes supérieures sur le nombre de repositionnement.

7.3 Comparaison des modèles

Malheureusement, nous ne pouvons pas comparer la performance de notre modèle aux performances d'autres modèles existants : soit ils traitent une problématique légèrement différente soit ils appliquent leur modèle sur d'autres instances. A défaut, nous comparons trois variantes de notre modèle binaire : le modèle binaire, le modèle binaire avec prétraitement et le modèle binaire avec prétraitement et coupes.

Les expérimentations montrent que le prétraitement améliore la performance du modèle considérablement. Les temps de résolutions diminuent et plus d'instances peuvent être résolues. En plus la relaxation linéaire du modèle avec prétraitement est beaucoup plus serrée que la relaxation linéaire du modèle sans prétraitement est et plus serrée que la borne inférieure présentée ci-dessus. Les coupes sont trop lâches et n'ont aucun effet ni sur la résolution du modèle binaire ni sur la solution de la relaxation linéaire.

Chapitre 8 : Approche de type branch and price

Ce chapitre présente notre approche de résolution de type branch and price. Nous présentons la décomposition du modèle binaire en un problème maître et un sous-problème. Puis nous proposons deux variantes d'un sous-problème énumératif et une procédure de branchement. Des expérimentations numériques évaluent la qualité de cette approche.

8.1 Génération de colonnes

Nous décidons de garder les contraintes physiques dans le modèle maître et des contraintes sur la faisabilité des colonnes dans le sous-problème.

Problème maître :	**Sous-problème :**
- au plus un conteneur par position	- au plus un conteneur par position
- pas de conteneur flottant	- pas de conteneur flottant
- enlèvement du conteneur t à la période t	- enlèvement du conteneur t à la période t
- continuité dans le temps	- l'ordre LIFO est respecté
- minimiser le nombre de relocations	- seulement les conteneurs bloquants sont repositionnés

Avec cette décomposition, une colonne représente une série de mouvements qui enlève le conteneur t à la période t et qui repositionne tous les conteneurs bloquants. La figure A.8 montre comment une colonne se compose d'un enlèvement et de repositionnements et comment elle modifie la configuration lorsqu'elle est appliquée.

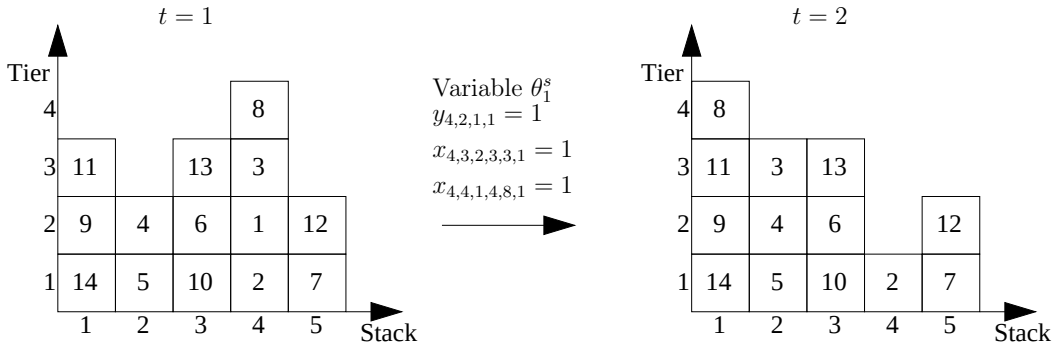


Figure A.8: La manière dont la colonne θ_1^s modifie la configuration (si appliquée)

Le problème maître est composé de versions adaptées des contraintes (7.16) à (7.20). A partir du problème maître dual nous déterminons le coût réduit d'une colonne qui est défini par

$$\sum_{i=1}^{W}\sum_{j=2}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{n=t+1}^{N} x^s_{ijklnt}\cdot(1+\gamma_{ijnt}-\gamma_{klnt})+\sum_{i=1}^{W}\sum_{j=1}^{H} y^s_{ijtt}\cdot\delta_{ijt}-\mu_t \tag{8.17}$$

où $\gamma_{ijnt} \in \mathbb{R}$ représente le bénéfice d'enlever le conteneur bloquant n de la position (i,j) à la période t, $\gamma_{klnt} \in \mathbb{R}$ le bénéfice de positionner le conteneur bloquant n dans la position (k,l) à la période t, δ_{ijt} le bénéfice d'enlever le conteneur t de la position (i,j) à la période t et μ_t le bénéfice de la période t.

Nous formulons un sous-problème binaire qui détermine pour chaque période t la colonne avec le coût réduit minimum tout en respectant toutes les contraintes sur la faisabilité de la colonne. De plus, nous proposons une nouvelle borne supérieure sur le nombre de relocations par période qui est basé sur la valeur des variables duales.

8.2 Sous-problème énumératif

Nous proposons un sous-problème qui énumère toutes les colonnes réalisables et rajoute celles avec un coût réduit négatif au problème maître. Nous utilisons différents mécanismes

pour réduire le nombre de colonnes à énumérer. Nous utilisons les bornes sur le nombre de relocations pour réduire le nombre de colonnes à générer. En plus, nous gardons trace des emplacements possibles pour chaque conteneur à chaque période. Nous ne générons pas toutes les colonnes, mais seulement celles qui peuvent être obtenues à partir des emplacements possibles.

Nous proposons une variante de cette approche où nous ne résolvons pas directement le problème de la période 1 à la période T. Au lieu de cela, nous itérons sur toutes les périodes : pour chaque période t nous résolvons le problème pour enlever les conteneurs 1 à t avec un minimum de repositionnements. Ceci nous permet de resserrer la borne supérieure et d'introduire une nouvelle condition d'optimalité.

8.3 Branch and price

Dans une solution fractionnaire un conteneur peut être placé dans plusieurs positions dans des piles différentes où dans la même pile. Nous proposons de brancher sur le conteneur avec la valeur de b_{ijnt} la plus proche de 0.5. Si le conteneur est placé dans différentes piles nous créons deux nœuds qui chacun interdisent de placer ce conteneur dans une partie des piles. Si le conteneur est placé dans la même pile nous créons deux nœuds qui chacun interdisent de placer le conteneur dans une partie des niveaux de cette pile.

8.4 Résultats expérimentaux

Nous évaluons la performance du branch and price (B&P) et de la génération de colonnes avec le sous-problème binaire (sBIP), avec le sous-problème énumératif (sEnum) et avec le sous-problème énumératif dans une approche itérative (sIter). Nous imposons une limite de temps de 60 minutes.

sBIP n'est pas très performant car trop de temps ($< 80\%$) est utilisé pour mettre à jour les valeurs des variables duales dans le sous-problème et peu de temps reste pour résoudre le problème. sEnum et sIter résolvent plus d'instances mais n'arrivent pas à résoudre les grandes instances car le nombre de variables à énumérer est beaucoup trop important. B&P obtient quelques solutions entières, mais n'est pas très efficace car les nœuds ne sont pas résolus rapidement.

Chapitre 9 : Approche de type branch and price heuristique

Ce chapitre présente notre approche de résolution de type branch and price heuristique. Nous utilisons le même problème maître que dans le chapitre précédent et détaillons un sous-problème heuristique et la stratégie de branchement. L'objective de cette approche est d'obtenir des bons résultats entiers plutôt que la solution fractionnaire optimale. Des expérimentations numériques évaluent la qualité de cette approche.

9.1 Sous-problème heuristique

Le sous-problème heuristique fonctionne en deux étapes : d'abord il détermine les conteneurs bloquants à enlever, puis il détermine où repositionner ces conteneurs. Nous utilisons des bornes sur le nombre de relocations et les emplacements possibles pour ne pas générer trop de colonnes qui ne peuvent jamais être dans la solution entière optimale.

Pour chaque période t, nous itérons sur toutes les position (i, j) où le conteneur à enlever t peut être placé. Pour chaque position nous déterminons les conteneurs bloquants à enlever – ceci est répété pour différents nombre de repositionnements. Cette décision est basée sur les emplacements possibles et sur les valeurs des variables duales. L'objectif est de déterminer les enlèvements des conteneurs bloquants qui sont le plus profitables à l'égard du coût réduit de la colonne. Nous représentons cette problématique comme un problème de flot à cout minimum et le résolvons par un programme linéaire.

Ensuite, nous déterminons où repositionner les conteneurs bloquants. Ceci est fait en fonction de la solution du problème maître primal, des valeurs des variables duales et des règles de repositionnement heuristiques et des contraintes de réalisabilité de la colonne. La solution obtenue par le sous-problème heuristique n'est pas forcement optimale, mais donne des bons résultats.

9.2 Branch and Price heuristique

Nous intégrons l'approche de génération de colonnes heuristique dans une approche de branch and price heuristique pour obtenir des solutions entières. L'idée est de réappliquer une heuristique gloutonne fréquemment pour obtenir des nouvelles solutions entières. Cette heuristique gloutonne est exécutée sur les configurations obtenues de la solution primale du problème maître et des colonnes avec un coût réduit négatif obtenues du sous-problème.

L'heuristique gloutonne ne peut être appliquée sur une période t que si la solution primale du problème maître est entière pour les périodes 1 à $t-1$. Nous branchons sur la première période non-entière t' et obtenons des nœuds avec des solutions entières jusqu'à la période $t'+1$. Nous utilisons les bornes inférieures et supérieures sur le nombre de relocations pour déterminer l'intérêt d'un nœud donné. Pour accélérer la résolution nous enlevons des colonnes du problème maître si leur nombre dépasse un certain seuil.

9.3 Résultat expérimentales

Nous comparons différentes variantes (différentes valeurs pour plusieurs paramètres) de notre branch and price entre elles et avec d'autres méthodes de résolution existantes. Notre meilleure variante est plus performante que certaines approches mais ils existent des approches encore plus performantes. Les résultats montrent que surtout pour les grandes instances beaucoup de temps ($< 90\%$) est nécessaire pour résoudre le problème maître. Par conséquent, le sous-problème et l'heuristique gloutonne sont exécutés très rarement et peu de solutions entières sont explorées. Nous sommes convaincus que notre approche obtiendra des meilleurs résultats lorsque le problème maître pourra être résolu plus rapidement.

Chapitre 10 : Version dynamique du problème de repositionnement de conteneurs

Dans ce chapitre, nous nous intéressons à la version dynamique du problème. Dans ce cas, l'ordre d'enlèvement des conteneurs n'est pas connu dès le début, mais révélé dans le temps. Nous présentons le problème et un indicateur pour décrire la qualité d'une configuration. Nous décrivons différentes stratégies de repositionnement et les comparons entre elles.

10.1 Description du problème et état de l'art

Les terminaux à conteneurs ont peu d'informations sur l'heure et l'ordre exact des arrivées de camions. ll n'est pas inhabituel qu'un terminal obtienne cette information uniquement lorsque les camions se présentent à l'entrée du terminal.

Nous supposons que les camions sont servis avec une politique FIFO. Dans ce cas, les temps de service dépendent principalement du nombre de repositionnements. Le terminal connait donc les prochains conteneurs à enlever mais n'a pas d'information sur les enlèvements suivants. La version dynamique du problème de repositionnement s'appuie sur les hypothèses A2 à A10.

- A2 –A8 : tel que présentées dans le chapitre 6
- A9: L'ordre d'enlèvement des conteneurs est dévoilé au fils du temps
- A10: Les camions sont sevis dans l'ordre FIFO.

10.2 Indicateur de qualité

Nous supposons que nous n'avons aucune information sur l'ordre des enlèvements. Dans ce cas, chaque conteneur peut être le prochain à être enlevé. Nous introduisons l'espérance du nombre de repositionnements (EVR) comme indicateur de qualité. Nous démontrons que l'EVR est minimale pour les configurations équilibrées où la différence entre la pile la plus haute et la plus basse est minimale.

10.3 Stratégies de repositionnement

Nous présentons différentes stratégies de repositionnement en cas de connaissance partielle sur les prochains D conteneurs á enlever.

- S1 : repositionner au hasard ;

- S2 : repositionner vers la pile la moins haute (ce qui minimise l'EVR de la configuration obtenue) ;

- S3 : minimiser le nombre de relocations pour enlever les prochains D conteneurs avec mise à jour en cas de nouvelles informations ;

- S4 : minimiser le nombre de relocations pour enlever les prochains D conteneurs sans mise à jour ;

- S5 : En plus des prochains D conteneurs à enlever, les D' conteneurs à enlever ensuite (mais pas leur ordre exact) sont supposés connus. Minimiser le nombre de relocations pour enlever les prochains D conteneurs et minimiser le nombre de conteneurs au-dessus des conteneurs D'

- S6 : Minimiser le nombre de relocations pour enlever les prochains D conteneurs et minimiser l'EVR de la nouvelle configuration

- S7 : Minimiser le nombre de relocations pour enlever les prochains D conteneurs et récompenser une pile vide dans la nouvelle configuration

- S7 : Les prochains D conteneurs peuvent être servis dans n'importe quel ordre : minimiser le nombre de relocations pour enlever les prochains D conteneurs

10.4 Résultats expérimentaux

Dans ce chapitre nous comparons les différentes stratégies entre elles et avec la solution optimale du problème statique si l'ordre d'enlèvement complet est connu dès le début. Les stratégies proposées obtiennent des meilleurs résultats que la stratégie heuristique, mais nécessitent plus de relocations que dans le cas statique. Surtout, les stratégies qui équilibrent la taille des piles obtiennent des bons résultats. Les temps de calcul sont assez courts pour pouvoir appliquer ces stratégies à un terminal.

Conclusion et perspectives

Les nouvelles technologies (échange de données, RFID et géolocalisation) surveillent et gèrent l'équipement et les flux d'informations. Les terminaux à conteneurs utilisent ces technologies pour échanger des données avec leurs partenaires, pour localiser les conteneurs et leurs équipements dans le terminal et pour automatiser des tâches. Cette thèse a démontré, à l'aide de deux exemples, comment ces informations peuvent être utilisées pour optimiser les opérations du terminal. Dans la première partie de cette thèse, nous avons utilisé les informations sur les dates d'arrivée et de départ annoncées et sur les volumes annoncés pour optimiser l'allocation des cavaliers. Dans la deuxième partie, nous avons utilisé les informations sur l'ordre d'enlèvement et l'emplacement des conteneurs afin d'améliorer le processus de déstockage. Ce chapitre résume les travaux exécutés et expose des perspectives pour continuer ce travail.

La première partie de cette thèse a abordé le problème d'affectation de cavaliers. L'objectif est d'allouer les cavaliers aux différents modes de transport (par exemple, camions, trains, barges et navires) de manière à minimiser le délai global du terminal. Cette problématique n'a pas été abordée dans la littérature précédemment. Les quatre principales contributions de cette thèse pour le problème d'allocation de cavalier sont les suivantes. Tout d'abord, nous avons proposé une notation pour décrire les différentes stratégies utilisées pour servir les différents modes de transport aux différents terminaux. Nous avons également déterminé la complexité de certaines stratégies de service. Deuxièmement, nous avons représenté le problème d'allocation de cavalier comme un problème de flot : les conteneurs à déplacer sont modélisés comme des flots et les cavaliers affectés comme capacités sur les arcs. Nous avons modélisé ce problème de flot comme un programme linéaire mixte en nombres entiers. Nous avons proposé une formulation générique et montré que le modèle générique peut facilement être adapté aux différentes stratégies de service. Troisièmement, nous avons effectué une étude de cas pour un terminal au Grand Port Maritime de Marseille. Nous avons montré par simulation que les résultats obtenus par le problème d'optimisation déterministe restent valable dans un environnement incertain. Quatrièmement, nous avons combiné le problème d'affectation de cavaliers avec le dimensionnement d'un système de rendez-vous pour camions. L'idée est d'utiliser le système de rendez-vous pour dévier les arrivées des camions vers les périodes creuses. Les expériences, menées avec un modèle d'optimisation déterministe adapté et un modèle de simulation, ont montré que cette approche combinée permet de réduire le délai global du terminal.

Pour poursuivre ce travail, le modèle d'optimisation peut être étendu pour représenter la situation du terminal plus en détail. Plusieurs types de conteneurs pourront être différenciés et représentés avec leurs caractéristiques spécifiques. En outre, la capacité de

manutention peut être représentée avec plus de détails, par exemple en incluant la congestion et l'emplacement des conteneurs dans le yard. Ceci permettrait aussi d'obtenir des informations sur les interactions entre l'affectation de cavaliers et l'affectation de la place de stockage. A cet effet, le modèle d'optimisation pourrait être combiné avec de la simulation ou des modèles de file d'attente. Il serait également intéressant d'inclure les aspects stochastiques dans le modèle d'optimisation ou de combiner le problème d'allocation de cavaliers avec la gestion des ressources humaines.

La deuxième partie de cette thèse a abordé le problème de repositionnement de conteneurs. L'objectif est d'enlever les conteneurs d'une rangée dans un ordre donné avec un nombre minimal de repositionnements parasites. Comme la plupart des études menées, nous avons considéré le cas où seulement les conteneurs au-dessus du conteneur à enlever peuvent être repositionnés. Les quatre principales contributions de cette thèse pour le problème de repositionnement de conteneurs sont les suivantes. Tout d'abord, nous avons amélioré un programme binaire existant. Ensuite, nous avons introduit une étape de prétraitement - qui améliore les performances du programme binaire considérablement - et deux bornes supérieures sur le nombre de repositionnements. Deuxièmement, nous avons présenté un algorithme de génération de colonnes pour ce problème. Nous avons formulé un sous-problème binaire et deux variantes d'un sous-problème énumératif. Nous avons intégré la génération de colonnes dans une démarche de type branch and price. Le branch and price résout des petites instances, mais n'obtient pas des résultats satisfaisants pour les grandes instances. Troisièmement, nous avons proposé un branch and price heuristique dont l'objectif est d'obtenir une bonne solution entière plutôt que la solution fractionnaire optimale. Cette approche utilise un sous-problème heuristique pour générer des nouvelles colonnes - basé sur les variables duales et des règles de repositionnement gloutonnes - et réapplique une heuristique pour obtenir des nouvelles solutions entières à plusieurs reprises. Le branch and price heuristique obtient des bons résultats pour certaines grandes instances, mais peut sans doute être amélioré. Enfin, nous avons abordé la version dynamique du problème de repositionnement de conteneurs où la séquence d'enlèvement est révélée au fil du temps. Nous avons introduit un critère pour évaluer la qualité d'une rangée et évalué différentes stratégies de repositionnement.

Ce travail peut être continué dans plusieurs directions. Pour une meilleure compréhension du problème, des règles de dominance pour les repositionnements pourront être déterminées. Ces règles pourront ensuite être utilisées pour obtenir des heuristiques plus performantes. Il serait intéressant de comparer la performance du modèle binaire utilisé ici avec des modèles linéaires existants. Une autre piste est de trouver des coupes pour le modèle binaire et le problème maître pour les rendre plus efficaces. Pour le branch and price exact, un sous-problème plus performant doit être trouvé. Différentes stratégies de branchement pourront ensuite être testées. Pour le branch and price heuristique, le meilleur réglage des paramètres doit être déterminé. En plus, le temps passé à résoudre le problème maître doit être réduit. La qualité de la solution globale peut être améliorée en améliorant l'heuristique utilisée. Pour la version dynamique du problème, une analyse plus théorique devra être effectuée pour déterminer les pires performances. Il est également possible d'évaluer différentes variantes du problème, par exemple permettre que toutes les conteneurs (et pas uniquement les conteneurs bloquants) puissent être repositionnés ou inclure les tâches de stockage. Pour la version dynamique, différents scénarios par rapport à la disponibilité et la fiabilité des informations pourront être analysés.

List of Figures

List of Tables

List of Algorithms

Bibliography

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993). *Network flows*, Prentice Hall.

Alessandri, A., Cervellera, C., Cuneo, M. and Gaggero, M. (2008). Nonlinear predictive control for the management of container flows in maritime intermodal terminals, *Proceedings of the 47th IEEE Conference on Decision and Control*, December 9-11, Cancún, Mexico, pp. 2800–2805.

Angeloudis, P. and Bell, M. G. H. (2011). A review of container terminal simulation models, *Maritime Policy & Management* **38**: 523–540.

Balev, S., Guinand, F., Lesauvage, G. and Olivier, D. (2009). Dynamical handling of straddle carriers activities on a container terminal in uncertain environment - a swarm intelligence approach, *Proceedings of the 3rd International Conference on Complex Systems and Applications*, June 29 - July 2, Le Havre, France.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**: 316–329.

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research* **202**: 615–627.

Bish, E. K., Chen, F. Y., Leong, Y. T., Nelson, B. L., Ng, J. W. C. and Simchi-Levi, D. (2005). Dispatching vehicles in a mega container terminal, *OR Spectrum* **27**: 491–506.

Borgman, B., van Asperen, E. and Dekker, R. (2010). Online rules for container stacking, *OR Spectrum* **32**: 687–716.

Borodin, A. and El-Yaniv, R. (1998). *Online computation and competitive analysis*, Cambridge University Press.

Böse, J., Reiners, T., Steenken, D. and Voß, S. (2000). Vehicle dispatching at seaport container terminals using evolutionary algorithms, *Proceedings of the 33rd Hawaii International Conference on System Sciences*, January 4-7, Maui, Hawaii.

Brinkmann, B. (2011). Operations systems of container terminals: a compendious overview, *in* J. Böse (ed.), *Handbook of Terminal Planning*, Springer New York, chapter 2, pp. 25–39.

Briskorn, D., Drexl, A. and Hartmann, S. (2006). Inventory-based dispatching of automated guided vehicles on container terminals, *OR Spectrum* **28**: 611–630.

Brucker, P. and Knust, S. (n.d.). The scheduling zoo - a searchable bibliography on scheduling, `http://www-desir.lip6.fr/~durrc/query/`.

Cao, J. X., Lee, D.-H., Chen, J. H. and Shi, Q. (2010). The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods, *Transportation Research Part E* **46**: 344–353.

Caserta, M., Schwarze, S. and Voß, S. (2009). A new binary description of the blocks relocation problem and benefits in a look ahead heuristic, *in* C. Cotta and P. Cowling (eds), *Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science*, Vol. 5482, Springer Berlin Heidelberg, pp. 37–48.

Caserta, M., Schwarze, S. and Voß, S. (2011). Container rehandling at maritime container terminals, *in* J. Böse (ed.), *Handbook of Terminal Planning Operations*, Vol. 49, Springer New York, chapter 13, pp. 247–269.

Caserta, M., Schwarze, S. and Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem, *European Journal of Operational Research* **219**: 96–104.

Caserta, M. and Voß, S. (2009). Corridor selection and fine tuning for the corridor method, *in* T. Stützle (ed.), *Learning and Intelligent Optimization, Lecture Notes in Computer Science*, Vol. 5851, Springer Berlin Heidelberg, pp. 163–175.

Caserta, M., Voß, S. and Sniedovich, M. (2011). Applying the corridor method to a blocks relocation problem, *OR Spectrum* **33**: 915–929.

Casey, B. and Kozan, E. (2012). Optimising container storage processes at multimodal terminals, *Journal of the Operational Research Society* **63**: 1126–1142.

Chen, G., Govindan, K. and Yang, Z. (2013). Managing truck arrivals with time windows to alleviate gate congestion at container terminals, *International Journal of Production Economics* **141**: 179–188.

Chen, L., Bostel, N., Dejax, P., Cai, J. and Xi, L. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, *European Journal of Operational Research* **181**: 40–58.

Chen, L., Langevin, A. and Lu, Z. (2013). Integrated scheduling of crane handling and truck transportation in a maritime container terminal, *European Journal of Operational Research* **225**: 142–152.

Chen, X., Zhou, X. and List, G. F. (2011). Using time-varying tolls to optimize truck arrivals at ports, *Transportation Research Part E* **47**: 965–982.

CMA-CGM (n.d.). CMA CGM Marco Polo, `http://www.cmacgm-marcopolo.com/fr/`, accessed January 2013.

Das, S. K. and Spasovic, L. (2003). Scheduling material handling vehicles in a container terminal, *Production Planning & Control* **14**: 623–633.

Dauzère-Pérès, S., Rouquet, A., Reynaud, C. and Zehendner, E. (2012). Conception et validation d'un outil de prévision des flux routiers d'un terminal portuaire à conteneur, *ATEC-ITS : Intelligence dans les déplacements*, February 1-2, Versailles, France.

Dekker, R., Voogd, P. and van Asperen, E. (2006). Advanced methods for container stacking, *OR Spectrum* **28**: 563–586.

Dempsey, M. (2011). RFID in ports and terminals, Information paper of the Port Equipment Manufactures Association (PEMA) `http://bit.ly/pemarfid`, accessed July 2013.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation, *in* G. Desaulniers, J. Desrosiers and M. M. Solomon (eds), *Column generation*, Springer US, chapter 1, pp. 1–32.

Economic Commission for Europe (2001). Terminology on combined transport, `http://www.unece.org/fileadmin/DAM/trans/wp24/documents/term.pdf`, accessed July 2013.

European Sea Ports Organisation (2010). Traffic data of year 2012 , `http://www.espo.be/index.php?option=com\_content\&view=article\&id=95\&Itemid=90`, accessed July 2013.

Fancello, G., Pani, C., Pisano, M., Serra, P., Zuddas, P. and Fadda, P. (2011). Prediction of arrival times and human resources allocation for container terminal, *Maritime Economics & Logistics* **13**: 142–173.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems, *4OR* **8**: 407–424.

Forster, F. and Bortfeldt, A. (2012). A tree search procedure for the container relocation problem, *Computers & Operations Research* **39**: 299–309.

Froyland, G., Koch, T., Megow, N., Duane, E. and Wren, H. (2008). Optimizing the landside operation of a container terminal, *OR Spectrum* **30**: 53–75.

Gambardella, L. M., Mastrolilli, M., Rizzoli, A. E. and Zaffalon, M. (2001). An optimization methodology for intermodal terminal management, *Journal of Intelligent Manufacturing* **12**: 521–534.

Gambardella, L. M., Rizzoli, A. E. and Zaffalon, M. (1998). Simulation and planning of an intermodal container terminal, *Simulation* **71**: 107–116.

Giuliano, G. and O'Brien, T. (2007). Reducing port-related truck emissions: The terminal gate appointment system at the Ports of Los Angeles and Long Beach, *Transportation Research Part D* **12**: 460–473.

Graham, R., Lawler, E., Lenstra, J. and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling theory: A survey, *Annals of Discrete Mathematics* **5**: 287–326.

Grötschel, M., Krumke, S. O., Rambau, J., Winter, T. and Zimmermann, U. T. (2001). Combinatorial online optimization in real time, *in* M. Grötschel, S. O. Krumke and J. Rambau (eds), *Online optimization of large scale systems*, Springer Berlin Heidelberg, pp. 679–704.

Guan, C. Q. and Liu, R. R. (2009). Modeling gate congestion of marine container terminals, truck waiting cost and optimization, *Transportation Research Record* **2100**: 58–67.

Hadjiconstantinou, E. and Ma, N. L. (2009). Evaluating straddle carrier deployment policies: a simulation study for the Piraeus container terminal, *Maritime Policy & Management* **36**: 353–366.

Hartmann, S. (2004). A general framework for scheduling equipment and manpower at container terminals, *OR Spectrum* **26**: 51–74.

Huynh, N. N. (2005). *Methodologies for reducing truck turn time at marine container terminals*, PhD thesis, The University of Texas at Austin.

Huynh, N. and Walton, C. M. (2008). Robust scheduling of truck arrivals at marine container terminals, *Journal of Transportation Engineering* **134**: 347–353.

Ioannou, P. A., Kosmatopoulos, E., Jula, H., Collinge, A., Liu, C.-I. Asef-Vaziri, A. and Doughtery, E. (2000). Cargo Handling Technologies, Prepared for the Center for Commercial Deployment of Transportation Technologies, `http://www.usc.edu/dept/ee/catt/assets/001/68518.pdf`, accessed July 2013.

Ioannou, P., Chassiakos, A. and Jula, H. (2006). Cooperative time window generation for cargo delivery / pick up with application to container terminals, Final Report METRANS Project 03-18, `http://www.metrans.org/research/final/13-18_Final.pdf`, accessed July 2013.

Ioannou, P. and Jula, H. (2008). Automated container terminal concepts, *in* P. A. Ioannou (ed.), *Intelligent freight transportation*, CRC Press, chapter 2, pp. 7–34.

Jaillet, P. and Wagner, M. R. (2010). Online optimization - an introduction, *in* J. Hasenbein, P. Gray and H. J. Greenberg (eds), *TutORials in Operations Research: Risk and Optimization in an Uncertain World*, INFORMS, chapter 6, pp. 142–152.

Jang, D.-W., Kim, S. W. and Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations, *International Journal of Production Economics* **143**: 256–262.

Kang, S., Medina, J. C. and Ouyang, Y. (2008). Optimal operations of transportation fleet for unloading activities at container ports, *Transportation Research Part B* **42**: 970–984.

Kelton, W. D., Sadowski, R. P. and Sturrock, D. T. (2007). *Simulation with Arena*, Mc Graw - Hill.

Kemme, N. (2013). Container terminal logistics, *Design and Operation of Automated Container Storage Systems*, Physica-Verlag HD, pp. 9–52.

Kim, K. H. (2008). Operational issues in modern container terminals, *in* P. A. Ioannou (ed.), *Intelligent Freight Transportation*, CRC Press, chapter 4, pp. 51–69.

Kim, K. H. and Bae, J. W. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals, *Transportation Science* **38**: 224–234.

Kim, K. H. and Hong, G.-P. (2006). A heuristic rule for relocating blocks, *Computers & Operations Research* **33**: 940–954.

Kim, K. H., Kim, K. W., Hwang, H. and Ko, C. S. (2004). Operator-scheduling using a constraint satisfaction technique in port container terminals, *Computers & Industrial Engineering* **46**: 373–381.

Kim, K. H., Lee, K. M. and Hwang, H. (2003). Sequencing delivery and receiving operations for yard cranes in port container terminals, *International Journal of Production Economics* **84**: 283–292.

Kim, K. Y. and Kim, K. H. (1999). A routing algorithm for a single straddle carrier to load export containers onto a containership, *International Journal of Production Economics* **59**: 425–433.

Koppe, B. and Brinkmann, B. (2008). State of the art of handling and storage systems on container terminals, *Chinese-German Joint Symposium on Hydraulic and Coastal Engineering*, August 24-30, Darmstadt, Germany.

Kozan, E. (2000). Optimising container transfers at multimodal terminals, *Mathematical and Computer Modelling* **31**: 235–243.

Kozan, E. and Preston, P. (2006). Mathematical modelling of container transfers and storage locations at seaport terminals, *OR Spectrum* **28**: 519–537.

Krumke, S. O. and Thielen, C. (2012). Introduction to online optimization, Lecture notes, University of Kaiserslautern, `http://optimierung.mathematik.uni-kl.de/groups/5/Lectures/OnlineWS1112/online-optimization.pdf`, accessed January 2013.

Lau, H. Y. and Zhao, Y. (2008). Integrated scheduling of handling equipment at automated container terminals, *International Journal of Production Economics* **112**: 665–682.

Lee, D.-H., Cao, J. X., Shi, Q. and Chen, J. H. (2009). A heuristic algorithm for yard truck scheduling and storage allocation problems, *Transportation Research Part E: Logistics and Transportation Review* **45**: 810–820.

Lee, L. H., Chew, E. P., Tan, K. C. and Wang, Y. (2010). Vehicle dispatching algorithms for container transshipment hubs, *OR Spectrum* **32**: 663–685.

Lee, Y. and Hsu, N. (2007). An optimization model for the container pre-marshalling problem, *Computers & Operations Research* **34**: 3295–3313.

Legato, P. and Monaco, M. F. (2003). Human resources management at a marine container terminal, *European Journal of Operational Research* **156**: 769–781.

Maguire, A., Ivey, S., Golias, M. and Lipinsk, M. (2010). Relieving congestion at intermodal marine container terminals: review of tactical/operational strategies, Annual Meeting of the Transportation Research Board, Washington, DC, USA, `http://www.trforum.org/forum/downloads/2010_161_Relieving_Congestion_Marine_Terminals_Strategies.pdf`, accessed July 2013.

Meersmans, P. J. and Wagelmans, A. P. (2001). Dynamic scheduling of handling equipment at automated container terminals, ERIM Report Series Reference No. ERS-2001-69-LIS, `http://repub.eur.nl/res/pub/137/erimrs20011213173918.pdf`, accessed July 2013.

Meisel, F. (2009). *Seaside operations planning in container terminals*, Physica-Verlag HD.

Morais, P. and Lord, E. (2006). Terminal appointment system study, Prepared for Transportation Development Centre of Transport Canada, `http://www.tc.gc.ca/media/documents/policy/14570e.pdf`, accessed July 2013.

Murty, K. G., Wan, Y.-W., Liu, J., Tseng, M. M., Leung, E., Lai, K.-K. and Chiu, H. W. C. (2005). Hongkong International Terminals gains elastic capacity using a data-intensive decision-support system, *Interfaces* **35**: 61–75.

Namboothiri, R. (2006). *Planning container drayage operations at congested seaports*, PhD thesis, Georgia Institute of Technology.

Namboothiri, R. and Erera, A. L. (2007). Planning local container drayage operations given a port access appointment system, *Transportation Research Part E* **44**: 185–202.

Nguyen, V. D. and Kim, K. H. (2009). A dispatching method for automated lifting vehicles in automated port container terminals, *Computers & Industrial Engineering* **56**: 1002–1020.

Nguyen, V. D. and Kim, K. H. (2010). Dispatching vehicles considering uncertain handling times at port container terminals, *in* K. Ellis, K. Gue, R. de Koster, R. Meller, B. Montreuil and M. Ogle (eds), *Progress in Material Handling Research, 11th International Material Handling Research Colloquium*, Material Handling Industry of America, Milwaukee, USA, June 21-24, 2012, pp. 210–226.

Notteboom, T. (2008). The relationship between seaports and the inter-modal hinterland in light of global supply chains, International Transport Forum, Discussion Paper No 2008-10, March, `http://www.internationaltransportforum.org/jtrc/discussionpapers/DP200810.pdf`, accessed July 2013.

Notteboom, T. and Winkelmans, W. (2004). Factual report on the European port sector, Report commissioned by European Sea Ports Organisation (ESPO), `http://www.rajaeeportandroadtraffic.com/RelatedDocs/Factual%20Report%20on%20European%20%20Port%20Sector.pdf`, accessed July 2013.

Park, T., Choe, R., Kim, Y. H. and Ryu, K. R. (2011). Dynamic adjustment of container stacking policy in an automated container terminal, *International Journal of Production Economics* **133**: 385–392.

Petering, M. E. and Hussein, M. I. (2013). A New Mixed Integer Program and Extended Look-Ahead Heuristic Algorithm for the Block Relocation Problem, *European Journal of Operational Research* **231**: 120–130.

Preston, P. and Kozan, E. (2001). An approach to determine storage locations of containers at seaport terminals, *Computers & Operations Research* **28**: 983–995.

Rei, R. and Pedroso, J. a. P. (2012). Tree search for the stacking problem, *Annals of Operations Research* **203**: 371–388.

Rodriguez Verjan, G. L. and Dauzère-Pérès, S. (2010). Optimisation des opérations en logistique portuaire. Master thesis in business engineering at l'Ecole des Mines de Saint-Etienne.

Sgouridis, S. and Angelides, D. (2002). Simulation-based analysis of handling inbound containers in a terminal, *Proceedings of the 2002 Winter Simulation Conference*, December 8-11, Los Alamitos, CA, USA, pp. 1716–1724.

Skinner, B., Yuan, S., Huang, S., Liu, D., Cai, B., Dissanayake, G., Lau, H., Bott, A. and Pagac, D. (2013). Optimisation for job scheduling at automated container terminals using genetic algorithm, *Computers & Industrial Engineering* **64**: 511–523.

Srour, J., Kennedy, J., Jensen, M. and Mitchell, C. (2003). Freight Information Real-time System for Transport (FIRST) - Evaluation final report, U.S. Department of Transportation, `http://ntl.bts.gov/lib/jpodocs/repts_te/13951/13951.pdf`, accessed July 2013.

Stahlbock, R. and Voß, S. (2008a). Operations research at container terminals: a literature update, *OR Spectrum* **30**: 1–52.

Stahlbock, R. and Voß, S. (2008b). Vehicle routing problems and container terminal operations - un update of research, *in* B. Golden, S. Raghavan and E. Wasil (eds), *The vehicle routing problem - latest advances and new challenges*, Operations Research Computer Science Interfaces Series, Vol. 43, Springer US, pp. 551–589.

Steenken, D., Voß, S. and Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review, *OR Spectrum* **26**: 3–49.

Tang, L., Zhao, R. and Liu, J. (2012). Models and algorithms for shuffling problems in steel plants, *Naval Research Logistics* **59**: 502–524.

Tsilingris, P. S., Psaraftis, H. N. and Lyridis, D. V. (2007). Radio frequency identification (RFID) technology in ocean container transport, *International Association of Maritime Economists Conference,*, July 4-6, Athens, Greece.

UNCTAD Secretariat (2010). Review of maritime transport 2010, `http://unctad.org/en/pages/PublicationArchive.aspx?publicationid=1708`, accessed July 2013.

UNCTAD Secretariat (2012). Review of maritime transport 2012, `http://unctad.org/en/pages/PublicationWebflyer.aspx?publicationid=380`, accessed July 2013.

Ünlüyurt, T. and Aydin, C. (2012). Improved rehandling strategies for the container retrieval process, *Journal of Advanced Transportation* **46**: 378–393.

van Hentenryck, P. and Bent, R. (2006). *Online stochastic combinatorial optimization*, MIT Press.

Vanderbeck, F. (2005). Implementing mixed integer column generation, *in* G. Desaulniers, J. Desrosiers and M. M. Solomon (eds), *Column generation*, Springer US, chapter 12, pp. 331–358.

Vis, I. F. (2006). A comparative analysis of storage and retrieval equipment at a container terminal, *International Journal of Production Economics* **103**: 680–693.

Vis, I. F. and de Koster, R. (2003). Transshipment of containers at a container terminal: An overview, *European Journal of Operational Research* **147**: 1–16.

Vis, I. F., de Koster, R. and Savelsbergh, M. W. P. (2005). Minimum vehicle fleet size under time-window constraints at a container terminal, *Transportation Science* **39**: 249–260.

Wasesa, M., Muhammad, I. H. and van Heck, E. (2011). Improving the container terminal performance by incorporating location synchronization module to the pre-notification protocol, Unpublished paper presented at the International Conference on Computational Logistics, Hamburg, Germany, September 19-22, 2011.

Wiese, J., Suhl, L. and Kliewer, N. (2011). Planning container terminal layouts considering equipment types and storage block design, *in* J. Böse (ed.), *Handbook of Terminal Planning*, Springer New York, chapter 12, pp. 219–245.

Wolfe, M. and Troup, K. (2005). The freight technology story: Intelligent freight technologies and their benefits , Report FHWA-HOP-05-030 for the US Department of Transportation, `http://ops.fhwa.dot.gov/freight/intermodal/freight_tech_story/freight_tech_story.htm`, accessed July 2013.

Woo, S.-H., Pettit, S. J., Kwak, D.-W. and Beresford, A. K. (2011). Seaport research: A structured literature review on methodological issues since the 1980s, *Transportation Research Part A* **45**: 667–685.

Wu, K.-C., Hernández, R. and Ting, C.-J. (2009). Applying tabu search for minimizing reshuffle operations at container yards, *Journal of the Eastern Asia Society for Transportation Studies* **8**.

Wu, K.-C. and Ting, C.-J. (2010). A beam search algorithm for minimizing reshuffle operations at container yards, *Proceedings of the International Conference on Logistics and Maritime Systems*, September 15 - 17, Busan, Korea, pp. 703–710.

Wu, Y., Luo, J., Zhang, D. and Dong, M. (2013). An integrated programming model for storage management and vehicle scheduling at container terminals, *Research in Transportation Economics* **42**: 13–27.

Yang, J. H. and Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems, *Journal of Intelligent Manufacturing* **17**: 453–463.

Zehendner, E. and Feillet, D. (2013). Benefits of a truck appointment system on the service quality of inland transport modes at a multimodal container terminal, *European Journal of Operational Research*, available online: http://dx.doi.org/10.1016/j.ejor.2013.07.005.

Zehendner, E., Feillet, D., Absi, N. and Dauzère-Pérès, S. (2011). Solving the resource allocation problem in a multimodal container terminal as a network flow problem, *in* J. W. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock and S. Voß(eds), *Computational Logistics, Lecture Notes in Computer Science*, Vol. 6971, Springer Berlin Heidelberg, pp. 341–353.

Zehendner, E., Rodriguez Verjan, G., Feillet, D., Absi, N. and Dauzère-Pérès, S. (2013). Optimized allocation of straddle carriers to reduce overall delays at multimodal container terminals, *Flexible Services and Manufacturing*, to appear (doi: 10.1007/s10696-013-9188-1).

Zeng, Q. and Yang, Z. (2009). Integrating simulation and optimization to schedule loading operations in container terminals, *Computers & Operations Research* **36**: 1935–1944.

Zhang, H., Guo, S., Zhu, W., Lim, A. and Cheang, B. (2010). An investigation of IDA* algorithms for the container relocation problem, *in* N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez and M. Ali (eds), *Trends in Applied Intelligent Systems, Lecture Notes in Computer Science*, Vol. 6096, Springer Berlin Heidelberg, pp. 31–40.

Zhao, W. and Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling, *Transportation Research Part E* **46**: 327–343.

Zhu, W., Qin, H., Lim, A. and Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem, *IEEE Transactions on Automation Science and Engineering* **9**: 710–722.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2013 EMSE 0714

Elisabeth ZEHENDNER

OPERATIONS MANAGEMENT AT CONTAINER TERMINALS USING ADVANCED INFORMATION TECHNOLOGIES

Speciality: Industrial Engineering

Keywords: Resource Allocation Problem, Network Flow Problem, Container Relocation Problem, Branch and Price, Dynamic Optimization

Abstract:

Container terminals use intelligent freight technologies (e.g., EDI, RFID and GPS) to exchange data with their partners, to locate containers and equipment within the terminal, and to automate tasks. This thesis illustrates, via two examples, how this data may be used to optimize operations at the terminal.

The first part uses information on announced volumes to allocate internal handling equipment. The objective is to minimize overall delays at the terminal. The problem is represented as a network flow problem and implemented as a linear mixed integer programming model. A case study for a terminal at the Grand Port Maritime de Marseille is carried out. We also show that combining the allocation problem with the dimensioning of a truck appointment system may reduce overall delays at the terminal.

The second part uses information on announced container retrievals and container positions to improve retrieval operations. The objective is to retrieve containers from a bay in a given sequence with a minimum number of parasite relocations. We improve an existing binary programming model and introduce an exact branch and price approach - with a binary subproblem and two variants of an enumerative subproblem - and a heuristic branch and price approach - with a heuristic subproblem. The exact approach solves only small instances; the heuristic approach performs well on several instances, but should be improved further. We also deal with a dynamic version of the problem where the retrieval order becomes revealed over time and evaluate different relocation strategies for this case.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2013 EMSE 0714

Elisabeth ZEHENDNER

GESTION DES OPÉRATIONS DANS LES TERMINAUX À CONTENEURS À L'AIDE DE TECHNOLOGIES DE L'INFORMATION AVANCÉES

Spécialité: Génie Industriel

Mots clefs : Allocation de Ressources, Problème de Flot, Repositionnement de Conteneurs, Branch and Price, Optimisation Dynamique

Résumé:

Les terminaux à conteneurs utilisent les nouvelles technologies (EDI, RFID et GPS) pour échanger des données avec leurs partenaires, pour localiser les conteneurs et leurs équipements dans le terminal, et pour automatiser des tâches. Dans cette thèse, nous montrons comment ces informations peuvent être utilisées dans la gestion des opérations.

La première partie utilise les informations sur les volumes annoncés pour affecter des ressources internes dans le but de minimiser le retard global au terminal. Nous représentons cette problématique à l'aide d'un problème de flot que nous implémentons comme programme linéaire mixte. Une étude de cas est réalisée pour un terminal du Grand Port Maritime de Marseille. En outre, nous combinons le problème d'affectation de ressources avec le dimensionnement d'un système de rendez-vous. Ceci permet de minimiser le retard global.

La deuxième partie utilise les informations sur les conteneurs à retirer et leurs emplacements pour optimiser le déstockage. Le but est de retirer tous les conteneurs d'une rangée en minimisant le nombre de repositionnements parasites. Nous améliorons un modèle binaire, proposons une approche exacte de type branch and price - avec un sous-problème binaire et deux variantes d'un sous-problème énumératif - et en dérivons une approche heuristique - avec un sous-problème heuristique. L'approche exacte ne résout que les petites instances ; l'approche heuristique obtient des résultats satisfaisants mais devra être améliorée. Nous nous intéressons aussi à la version dynamique du problème où les informations sur les conteneurs à retirer arrivent petit à petit et comparons différentes stratégies de repositionnement.